VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Návrh a implementace modulu pro kapacitní plánování výroby v systému iMPROVE iT!

Design and Implementation Module for Capacity Planning in iMPROVE iT! System

2016 Bc. Lukáš Lindovský

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Lukáš Lindovský**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Návrh a implementace modulu pro kapacitní plánování výroby v systému iMPROVE iT!
Design and Implementation Module for Capacity Planning in iMPROVE iT! System

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je navrhnout a vytvořit softwarovou komponentu (modul) realizující kapacitní plánování výroby. Tato komponenta bude implementována ve frameworku existujícího výrobního informačního systému iMPROVE iT!. Tento modul bude na základě definic výrobních kapacit, technologických postupů a uživatelsky definované parametrizace zobrazovat optimální plán výroby. Součástí modulu bude uživatelské rozhraní pro zobrazení vypočteného výrobního plánu. Výsledný plán výroby bude také možno uživatelsky upravovat, čímž dojde k přeplánování a vzniku nové verze výrobního plánu.

1. Nastudování problematiky kapacitního plánování.
2. Nalezení a popis vhodných algoritmů využitelných pro kapacitní plánování.
3. Návrh a implementace modulu pro kapacitní plánování.
4. Integrace tohoto modulu do existujícího systému iMPROVE iT!
5. Testování modulu na reálných datech.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Petr Zima**

Konzultant diplomové práce: doc. Mgr. Miloš Kudělka, Ph.D.

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 26. dubna 2016 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 26. dubna 2016 .....

SCADA Servis s.r.o.
Čedičová 1378/6
710 00 Ostrava-Slezská Ostrava
IČ: 26791285, DIČ: CZ26791285

Rád bych na tomto místě poděkoval vedoucímu práce Ing. Petrovi Zimovi a konzultantovi práce doc. Mgr. Miloši Kudělkovi, Ph.D., kteří mi s prací pomohli, protože bez nich by tato práce nevznikla. V neposlední řadě také děkuji celé firmě SCADA Servis, ve které mi bylo umožněno tuto práci vytvořit.

Abstrakt

Tato diplomová práce se zabývá tvorbou modulu pro kapacitní plánování v systému iMPROVE iT!. První část práce pojednává o problematice optimalizace výrobních rozvrhů a popisuje algoritmy řešící tento problém. Dále je vytvořen návrh a implementace modulu s konkrétním vybraným algoritmem. Zároveň je vytvořen modul, který zajišťuje interaktivní uživatelské rozhraní pro zobrazení výrobního rozvrhu. Před samotným testováním jsou tyto moduly propojeny s dalšími potřebnými komponentami sytému iMPROVE iT!. Do tohoto systému jsou následně oba moduly integrovány. Poslední část práce je věnována testování a porovnávání výsledků přímo v sytému iMPROVE iT! s reálnými daty.

Klíčová slova: iMPROVE iT!, kapacitní plánování, výrobní rozvrh, optimalizace výrobního rozvrhu, rozvrhování operací, diferenciální evoluce

Abstract

This master's thesis deals with the creation of capacity planning module in iMPROVE iT! system. The first part is introduces question of optimizing production schedules, descriptions of algorithms solving this problem. In the next part module design and implementation are created with choosen algorithm. Simultaneously second module which provides an interactive user interface to display the production schedule has been created. Before testing these modules were linked with other necessary components of the iMPROVE iT! system. Into this system both modules has been integrated. The last part is dedicated to the testing and comparing the results directly in the system iMPROVE iT! with the real data.

Key Words: iMPROVE iT!, capacity planning, production schedule, production schedule optimization, operation scheduling, differential evolution

Obsah

Seznam použitých zkratek a symbolů

ACO	–	Ant Colony Optimization
APS	–	Advanced Planning and Scheduling
DE	–	Diferenciální Evoluce
ERP	–	Enterprise Resource Planning
ISA	–	The International Society of Automation
MES	–	Manufacturing Execution System
MOM	–	Manufacturing Operations Management
MRPI	–	Material Requirements Planning
MRPII	–	Manufacturing Resource Planning
ORM	–	Object relationship mapping
PSO	–	Particle Swarm Optimization
SA	–	Simulated Annealing
TS	–	Tabu Search
UML	–	Unified Modeling Language
XAF	–	eXpressApp Framework

Seznam obrázků

Seznam tabulek

Seznam algoritmů

1 Úvod

Velkým fenoménem u výrobních podniků v současné době je snaha co nejvíce zvýšit efektivitu výroby a snížit náklady na výrobu. Pokud chce být výrobní podnik konkurenceschopný, pak musí vyrábět efektivně a také hlavně vyrábět zboží, které zákazník požaduje a to ve slíbené dodací lhůtě. Pro tyto aspekty hraje významnou roli kvalita výrobního rozvrhu.

Problém rozvrhování je definován množinou zakázek, které chceme vyrobit a dodat zákazníkovi. Každá taková zakázka se skládá z několika operací, jejichž pořadí je dáno neměnným technologickým postupem. Každá určitá operace musí být prováděna na některém stroji z množiny možných strojů pro tuto operaci. Cílem je vytvoření takového rozvrhu, ve kterém bude každé operaci přidělen konkrétní stroj a bude dodržen technologický postup. Zároveň je nutné, aby byl rozvrh optimalizovaný dle určitého optimalizačního kritéria. Mezi nejčastější optimalizační kritérium patří celková doba trvání zakázek v rozvrhu. Cílem této optimalizace je co nejvíce snížit čas výroby těchto zakázek. Každá ušetřená hodina totiž podniku přináší zisk v podobě financí i cenného času navíc. Mezi další optimalizační kritéria patří upřednostňování prioritních zakázek před ty méně důležité nebo dodržování lhůt dokončení zakázek. Tyto optimalizační problémy rozvrhování se řadí mezi NP-těžké problémy, které ve větším rozsahu nejsou řešitelné exaktními algoritmy.

Tato práce se věnuje vytvoření modulu kapacitního plánování pro MES/MOM systém iMPROVE iT!. Nezbytným krokem při vytváření tohoto modulu bylo nastudování netriviálního problému a definování přesného cíle, který má tento modul řešit. Po detailním nastudování problematiky kapacitního plánování podle požadavků na tento modul byla upřesněna charakteristika cíle této práce, kterou nejlépe vystihuje pojem rozvrhování operací výroby. Řešená optimalizační kritéria jsou celkový čas výroby, priority zakázek, maximální zpoždění a maximální doba trvání zakázek ve výrobě. K řešení tohoto problému jsem zvolil algoritmus diferenciální evoluce. Konkrétní byznys problém, který je modulem řešen, byl definován jako rozvrhování zakázkové výroby s paralelními stroji (flexible job shop). Výsledný rozvrh je reprezentován strukturou disjunktivního grafu. Před nalezením kritické cesty, která definuje celkovou dobu trvání všech zakázek, je potřeba tento graf topologicky uspořádat. Pro interaktivní uživatelské rozhraní a zobrazení výsledného rozvrhu uživateli byl vytvořen druhý modul, který řeší tuto problematiku odděleně od samotného rozvrhování. Následně jsou oba tyto moduly integrovány do systému iMPROVE iT! a testovány nad reálnými daty. Při konverzi dat z tohoto systému a zpět do něj jsou využívány struktury postavené na normě ANSI/ISA 95, což je mezinárodní standard pro integraci podnikových a výrobních systémů.

Úvod to problematiky kapacitního plánování popisuje druhá kapitola práce. Mimo jiné je tato kapitola věnována představení a porovnání některých systémů plánování od jejich historie až po současnost. Kapitola třetí se věnuje kapacitnímu plánování hlouběji z hlediska základních pojmů. Jsou zde objasněny různé parametry tohoto problému, jednotlivé typy problémů rozvrhování, typy rozvrhů a specifikace problému v kontextu výpočetní složitosti. Ve 4. kapitole jsou

představeny vhodné algoritmy problému rozvrhování výroby. Konkrétně se jedná o algoritmy exaktní, heuristické, genetické či evoluční nebo algoritmy s prvky umělé inteligence. Kapitola pátá je již věnována přímo návrhu samotného modulu. Popisuje také vybraný algoritmus, reprezentaci výrobního rozvrhu a operace s ním spojené. Dále jsou zde popsány definované účelové funkce, které modul využívá při optimalizaci rozvrhů. Uživatelské rozhraní bylo implementováno ve zvláštním modulu, jehož funkce jsou uvedeny v šesté kapitole. Oba tyto moduly jsou následně integrovány do systému iMPROVE iT!, kde využívají struktur postavených na normě ANSI/ISA 95. Z těchto struktur jsou data převedena do struktur daných modulem kapacitního plánování. Úspěšné převedení dat mezi těmito strukturami je precedenční podmínkou spuštění algoritmu pro rozvrhování. Testování vstupních parametrů algoritmu diferenciální evoluce a daných účelových funkcí jsou v kapitole číslo sedm. Tento text je veřejnou částí práce a je nutno zdůraznit, že tato práce má také neveřejnou část.

2 Pojem kapacitní plánování

Kapitola obsahuje úvod do problematiky kapacitního plánování spolu s představením a následným porovnáním existujících systémů zabývajících se tímto tématem.

2.1 Prostředí kapacitního plánování výroby

Prvním důležitým krokem je vysvětlení, co znamená pojem kapacitní plánování výroby. Při snaze pochopit tento pojem, je potřeba se na proces výroby podívat z několika úhlů pohledu. Důležité je si uvědomit, jaká výrobní strategie je využívána, zda se vyrábí na zakázku nebo na sklad. Dalšími faktory mohou být různé kapacity ve výrobě. Nejčastěji se jedná o kapacity, kolik toho může daný stroj za určitou časovou jednotku zvládnout, co je potřeba k tomu, aby to zvládl, ať už se jedná o kapacity personální, materiál či jiné zdroje.

Metoda kapacitního plánování se využívá pro správné určení času nutného pro výrobu určitého výrobku. Žádoucí je, aby v procesu výroby byl optimální průtok materiálů a polotovarů od samotného začátku až po finální výrobek. Kapacitní pohled na celý proces výroby znamená, že je potřeba zvažovat kapacity konkrétních pracovišť, které může tvořit jeden konkrétní stroj, nebo také obyčejné nářadí, kterého je na daném pracovišti omezené množství.

2.2 Cíle a význam kapacitního plánování

Hlavním cílem strategického plánování kapacit je dosažení optimální úrovně, kde výrobní schopnosti uspokojí poptávku zákazníků. Pojem kapacity rozumíme potřebné nástroje, lidi a jejich schopnosti požadované k výrobě produktů. Pokud výrobní schopnosti nesplňují požadavky poptávky, nebo pokud jejich náklady na výrobu jsou příliš vysoké, může to mít důsledek přemáhání strojů či ztráty zákazníka.

Pojem kapacita lze chápat jako potenciál výroby zboží nebo poskytování služeb v průběhu určitého časového intervalu. Plánování kapacit může probíhat z dlouhodobého nebo krátkodobého hlediska. Dlouhodobé úvahy se týkají celkových úrovní kapacit zdrojů. Krátkodobé plánování zachytává momentální kapacitní změny v důsledku sezonních, náhodných a nepravidelných výkyvů v poptávce.

Nadbytečná kapacita vzniká v případě, kdy je skutečná výroba menší, než jaké je možné skutečně dosáhnout. Tato situace samozřejmě není pro výrobní podnik optimální, jelikož to znamená, že poptávka po výrobcích je nižší, než kterou by výrobní podnik byl schopen pro trh dodávat.

2.2.1 Kapacitní plánování a 4. průmyslová revoluce

4. průmyslová revoluce (Industry 4.0) je založena na konceptu kyberneticko-fyzikálních systémů, internetu věcí a internetu služeb. Cílem 4. průmyslové revoluce je plně automatizovat výrobní a rozhodovací procesy.

Mluvíme o horizontální a vertikální integraci. Horizontální integrací se rozumí integrace počínající objednávkou, výzkumných a vývojových procesů, plánování a rozvrhování výroby až po distribuční logistiku. S pojmem vertikální integrace se spojuje integrace od úrovně řízení v reálném čase, přes plánování a rozvrhování, ERP systémy až k rozhodování na nejvyšší úrovni vedení. Inteligentní plánování a rozvrhování leží na průsečíku horizontální a vertikální integrace v samotném srdci všech procesů Industry 4.0. [16]

Jedná se tedy o velmi důležitý a citlivý proces pro plně fungující Industry 4.0.

2.3 Rozvrhování

Rozvrhování je posledním krokem při plánování a návrhu provozu. Zabývá se výrobou potřebného množství produktu v požadovaném časovém intervalu a je prováděno z hlediska krátkodobého plánování. Rozvrhování pracovních míst se zabývá přidělováním zdrojů k prioritním úlohám s nejbližšími termíny dodání. Cílem vysoké efektivity rozvrhování je co nejlepší přiřazování pracovních sil a nástrojů tak, aby nastala rovnováha mezi časem a využíváním omezených zdrojů v rámci organizace. Při rozvrhování operací se provádí:

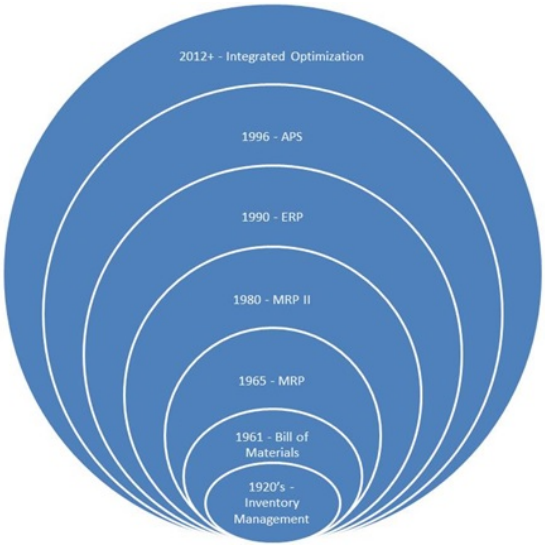
- Alokace zdrojů
- Rozvrhování pracovní síly
- Rozvrhování využití výrobních nástrojů

2.4 Historie kapacitního plánování

Pojem kapacitní plánování je známý již z padesátých let 19. století. Už v této době výrobní podniky realizovaly objednávání materiálu pomocí spotřebních prognóz, které získávaly na základě využití skladových karet. Avšak výsledek byl takový, že podniky ve většině případů držely zásoby nepřiměřené svým potřebám. Častým problémem bylo, že některé součástky se sice již nevyráběly, avšak díky své vysoké kapacitě na skladě se stále musely používat a jejich prodej byl vynucený.

MRP I

První systémy označované jako MRP se objevily v 60. letech minulého století. Princip fungování byl takový, že vedení podniku stanovilo hlavní výrobní plán, který sloužil jako vstup do MRP systému. Ten dokázal vypočítat množství materiálu, které je potřeba objednat. Takto MRP vytvořil seznam položek potřebných pro splnění daného plánu výroby. MRP systém využíval



Obrázek 1: Vývoj systémů kapacitního plánování výroby, převzato z [15]

data z kusovníku, stavu skladů a technologických postupů. Systém vycházel z předpokladů, že je pevně stanovena a známa průběžná doba výroby, která se již nebude měnit. Výsledný rozvrh tohoto systému nezohledňoval kapacitní omezení výrobních zařízení.

MRP II

V průběhu 70.let byla metoda MRP I postupně tak vylepšována, že v období 80.let světlo světa spatřil nový systém MRP II. Tento systém měl nahradit systém předešlý hlavně z toho důvodu, že při plánování bral v úvahu kapacitní a materiálové omezení, čímž byla kvalita výsledného plánu mnohem vyšší. Systém tedy rozplánoval výrobu podle zadaných požadavků od počátečního termínu dopředu nebo od finálního termínu směrem dozadu. Ačkoli systém MRP II byl lepší než jeho předchůdce, tak se přesto ukázalo, že se stále nejedná o ideální plánovací systém. Skutečné řízení se nedalo použít realisticky, protože systém předpokládal neomezenou dostupnost kapacit v čase i v době jejich potřeby.

Systémy MRP často bývají součástí ERP systémů, neznamená to však, že by nemohly fungovat samostatně na vlastní pěst. Nevýhodou může být skutečnost, že tyto systémy spoléhají na ruční zadání dat od uživatele potřebných k vytvoření výrobního plánu.

ERP

Systémy ERP jsou integrovány v rámci celé firmy a automatizují všechny procesy podniku. Hlavní myšlenka systému ERP byla, že bude jedna databáze, která bude obsahovat všechna data: informace o výrobě, distribuce, finance, lidské zdroje, nákup a sklad. Realita je taková, že většina ERP systémů vychází z principu MRP II. Přestože některé systémy podporují i další metody (např. Kanban, teorie omezení či simulační metody), patří mezi hlavní nevýhody ERP

systémů zejména plánování do neomezených kapacit a nízká schopnost reagovat na aktuální situaci v dílně.[2]

APS

Systémy APS se začaly objevovat v druhé polovině 90.let. Tento nový systém vznikl, aby bylo dosaženo lepšího plánování než poskytují plánovací systémy v rámci ERP. Bohužel k tomuto termínu se váže spoustu definicí. Nikde není přesně definováno, co musí systém APS obsahovat, aby mohl být jako APS systém klasifikován. Jedna z možných definic by mohla být následující:

„APS zahrnuje celý rozsah funkcí od operativního rozvrhování výroby se zohledněním všech úzkých (kritických) míst přes sledování plnění plánu, simulaci "co když", nástroje pro prognózování a slibování termínu dodávky až po aplikace pro řízení celých dodavatelských řetězců, jež se na trhu objevují v posledních měsících.“ [1]

APS zahrnuje kromě funkčnosti MRP II také schopnosti optimalizovat výrobní plán a určit pořadí, aby byla práce efektivní z hlediska seřizování strojů. Další žádoucí schopností je přizpůsobení plánu, ve kterém existuje několik omezení (např. nástroje a pracovní síla) s možností simulovat změny. APS systém je tedy vhodný pro společnosti s vysokým počtem objednávek, kde pořadí operací hraje důležitou roli.

Tento systém může být stejně jako systém MRP integrován do systému ERP. Z tabulky 1 lze vyčíst rozdíly mezi APS a MRP II systémy.

APS systém	**MRP II systém**
Přednost zákazníka se může měnit v závislosti na důležitosti zákazníka	Všichni zákazníci mají stejnou prioritu/přednost v sytému
Dodací lhůty lze dynamicky zadat kontaktováním zákazníků	Dodací lhůty jsou známé a stabilní
Dynamicky počítá plán a časový rozvrh během pár minut každou změnou	Průběh je typicky dán šarží a na delší horizont
Podpora lepšího rozhodování díky „co když“ analýze a simulaci	Nepodporuje žádné pomůcky pro rozhodování
Snadný přechod na nižší úroveň podávání zpráv založených na identifikaci mimořádných podmínek	Podrobné informace je obtížné vyčíst a rozluštit
Přidělování materiálů v závislosti na dostupnosti a zadaných kritérií	Přidělování materiálů prováděno podle toho, kdo dříve přijde

Tabulka 1: Rozdíly mezi APS a MRP II systémy - převzato z [4]

Oblast	**ERP systém**	**APS systém**
Filosofie plánování	- Plánování bez ohledu na omezenou dostupnost klíčových zdrojů potřebných pro realizaci plánů - Proveditelné plány - Tlakem - Sekvenční shora dolů	- Plánování poskytuje možné a únosné plány založené na omezené dostupnosti zdrojů - Optimalizované plány - Tahem - Integrované a souběžné
Podnikové řízení	Koordinace výroby	Uspokojení poptávky ze strany spotřebitelů
Rozsah výroby	Výhradně diskrétní výroba	Všechny typy výroby
Hlavní oblast podpory	Transakce: Finance, Kontrola, Výroba	Plánování: Poptávka, výroba, logistika, dodavatelský řetězec
Informační tok	Shora dolů	Obousměrný
Simulační schopnosti	Nízké	Vysoké
Schopnost optimalizovat náklady, cenu a zisk	Není k dispozici	Dostupné
Výrobní dodací lhůty	Pevné	Flexibilní
Inkrementální plánování schopnosti	Není k dispozici	Dostupné
Rychlost přeplánování	Nízká	Vysoká
Skladování dat a jejich výpočet	Databáze	Paměť

Tabulka 2: Srovnání APS systémů a ERP systémů - převzato z [4]

3 Problematika kapacitního plánování

V této kapitole je hlouběji rozebrána problematika kapacitního plánování.

3.1 Základní pojmy

V problematice kapacitního plánování se setkáváme s několika důležitými pojmy, kterým je potřeba rozumět z důvodu správného chápání tohoto problému. Mezi základní pojmy patří operace, úloha a stroj.

- Operace/aktivita (operation) je proveditelná technologická část výrobního procesu, která již dále není dělitelná na další části.
- Úloha/Zakázka (Job) je výrobní postup neboli zakázka, která se skládá z posloupnosti operací potřebných vykonat.
- Stroj/zdroj (machine) je zařízení, na kterém lze vykonat jednu či více operací.

Plánování a rozvrhování

V oblasti plánování a rozvrhování často bývají nejasnosti v rozdílech mezi těmito dvěma termíny. Pokud se bavíme o termínu plánování (timetabling), pak se jedná o orientační plánování, od kdy do kdy bude daná zakázka vykonána. Oproti tomu, úkolem rozvrhování (scheduling) je rozvrhnout dané zakázky či operace nejen do konkrétního času, ale také z hlediska konkrétních požadovaných zdrojů. Výsledkem plánování je výrobní plán, který udává, co a jak se má vyrobit (materiál, nástroje, strojní zařízení, personál a lhůty). Výsledkem rozvrhování je výrobní rozvrh, podle kterého lze určit, na kterém zdroji se operace bude provádět a také je známý začátek a konec provádění operace.

Logistika výroby v reálném světě obvykle začíná fází plánování, která určí zakázky, které se budou vyrábět. Druhým krokem je fáze rozvrhování, která určí přesné časy vykonávání operací a konkrétní zdroje.

Výrobní rozvrh

Výrobní rozvrh je množina operací, které jsou zařazené v čase a je jim přiřazený určitý zdroj tak, aby bylo dosaženo stanovených cílů. Další podmínkou rozvrhu je, aby byly splněny všechny podmínky, které jsou ke zdrojům definovány. Z hlediska výrobního rozvrhu můžeme zdroje chápat jako potřebné stroje. K těmto strojům jsou přiřazené operace, u kterých je jasně zřetelné od kdy do kdy mají trvat - být vykonávány. Zdrojem však nemusí být pouze stroj. Může se jednat o personál, který by měl tuto operaci vykonávat, nebo by na ni měl dohlížet. Cílem rozvrhu

by také mělo být zachytit jednotlivé omezení mezi operacemi. Mělo by tedy být jasně vidět, že operace B nesmí začít dříve, než skončí operace A. Rozvrh nám také udává alokaci těchto zdrojů pro dané operace. Pokud má být operace A vykonávána na stroji X ve středu 20:00 - 20:45, pak je jasné, že stroj X je v tuto dobu rezervovaný pro danou operaci A. Pro tento stroj tedy nelze v toto období plánovat žádné jiné operace. V neposlední řadě je potřeba počítat s kapacitami jednotlivých zdrojů, které dávají informaci o tom, co stroj umí vykonat a v jakém maximálním množství. Mezi nejčastěji používaný nástroj pro vizuální zobrazení rozvrhu je tzv. Ganttův diagram. Jedná se o diagram, kde ve svislé ose jsou zobrazeny jednotlivé stroje a vodorovná osa značí čas.

Statické parametry úlohy

Statickými parametry můžeme nazvat ty parametry, které nejsou jakkoliv závislé na výrobním rozvrhu.

- Doba trvání p_{ij} (processing time) - Doba trvání zpracování operace j na daném stroji i.
- Termín dostupnosti r_j (release date) - Je termín, kdy nejdříve můžeme zakázku j začít vykonávat na stroji i.
- Termín dokončení d_j (due date) - Je závazné datum dokončení zakázky j. Toto datum představuje termín, kdy by zakázka měla být dokončená a připravená k předání zákazníkovi.
- Váha w_j (weight) představuje prioritu pro výrobní zakázku vzhledem k ostatním zakázkám.

Dynamické parametry úlohy

Dynamické parametry jsou závislé na sestaveném rozvrhu.

- Počáteční datum S_{ij} (starting time) - Představuje datum a čas, kdy bude úloha j zahájena na stroji i
- Datum ukončení C_{ij} (completion date) - Je datum a čas dokončení úlohy j na stroji i.

Omezující podmínky

Při rozvrhování operací existuje celá řada omezujících podmínek, které musí být splněny, aby rozvrh byl rozvrhem korektním a bylo dosaženo všech požadovaných cílů. Mezi nejčastější omezující podmínky patří:

- Nastavovací čas strojů (setup time)

- Dostupnost materiálu
- Technologický postup
- Směnnost personálu
- Vhodnost stroje

V případě nastavovacího času strojů se jedná o čas, který je potřeba, aby byl stroj přenastaven/připraven při přechodu z jedné operace na druhou. Vezmeme si příklad, že máme lakovací linku, která bude provádět operaci A (lakování na červeno) a poté operaci B (lakování na modro). Mezi těmito dvěma operacemi je potřeba provést přenastavení stroje, což v tomto případě může znamenat výměnu červeného laku za modrý. Tento proces však zabere určitý čas (setup time), a pokud tato situace nastane, je potřeba s tímto časem při rozvrhování počítat.

Druhým nejčastějším případem omezujících podmínek je dostupnost materiálu. Pokud není potřebný materiál dostupný, nemůže být v daný moment operace využívající tento materiál vykonána.

Technologický postup je dalším důležitým faktorem při rozvrhování. Je potřeba vždy dodržet přesné pořadí operací, které definuje výrobní postup. Budeme-li mít zakázku na výrobu chleba, mohl by být technologický postup např. 1.míchání těsta, 2.pečení a 3.balení chleba v tomto pořadí. Tento postup také zahrnuje jednotlivé časové rezervy mezi operacemi. V tomto případě by mohla být časová rezerva např. 2 hodiny mezi pečením a balením chleba z důvodu, aby před balením chleba stihl vychladnout.

Omezující podmínka směnnosti personálu udává, že je potřeba vytvořit takový rozvrh, ve kterém bude vždy potřebný pracovník v požadovaný čas k dispozici. Směnnost personálu nám udává rozvrh směn personálu, ale velkou roli můžou hrát také dovolené či náhlé onemocnění pracovníků.

Posledním nejčastějším omezením bývá vhodnost stroje. Tím se chápe, že máme pro konkrétní operaci více alternativních strojů, na kterých může být tato operace vykonána. Cílem poté je rozvrhnout tuto operaci na stroj, který je nejvíce vhodný ze všech těchto alternativních strojů. V reálném případě může jít např. o to, že některý ze strojů může mít až několikanásobně vyšší výdaje za provedení stejné operace než ostatní stroje. Druhým případem by mohlo být, že určitý stroj stihne operaci 2x rychleji než jiný stroj.

3.2 Grahamova klasifikace

V oblasti plánování a rozvrhování se velice často používá termín Grahamova klasifikace, která umožňuje popisovat problémy rozvrhování. Zápis Grahamovy klasifikace 1 se skládá ze tří parametrů.

$$\alpha|\beta|\gamma \tag{1}$$

V tomto zápise každý parametr představuje popis různého odvětví problému.

- α - Popisuje charakteristiku strojů. Přesněji řečeno jakým způsobem budou úlohy na stroje alokovány.
- β - Popisuje charakteristiku úloh neboli omezení aplikované na úlohy.
- γ - Optimalizační kritéria

3.2.1 Charakteristika strojů

Položka α, která charakterizuje úlohu týkající se struktury strojů, může nabývat několika z níže uvedených hodnot.

Jeden stroj (1)

Jedná se o nejjednodušší případ, kde ve výrobě figuruje jediný stroj. V reálném prostředí se s tímto případem moc nesetkáme.

paralelní stroje (P_m)

Toto je charakteristika, kde se vyskytuje m stejných paralelních strojů. Pak úloha j skládající se z jedné operace může být provedena na kterémkoliv z m strojů.

Paralelní stroje s různou rychlostí (Q_m)

V tomto případě máme m paralelních strojů, kde každý z nich však pracuje s rozdílnou rychlostí. Doba trvání úlohy j na stroji i je přímo závislá na rychlosti v_i. Doba trvání je odvozena pomocí $p_{ij} = p_j/v_i$

Nezávislé paralelní stroje s různou rychlostí (R_m)

V tomto případě máme m paralelních strojů, avšak každý z nich umí vykonat danou úlohu různě rychle. Doba trvání úlohy j na stroji i je přímo závislá na rychlosti v_ij. Doba trvání je odvozena pomocí $p_{ij} = p_j/v_{ij}$

Multi-operační problémy (Shop)

Pokud máme výrobní postup, kde se daná úloha skládá z několika operací, které musí postupně projít několika stroji, pak mluvíme o tzv. multi-operačním problému. V tomto případě máme úlohu j, která se skládá z i operací. Každá operace o_{ij} je prováděna na stroji i a zabere dobu p_{ij}.

Výše uvedené případy byly zmíněny pouze zkratkovitě. Existují ještě další multi-operační problémy (Shop), které budou detailněji zmíněny v podkapitole 3.3. Konkrétně se jedná o Job Shop[1], Open Shop[1], Flow Shop[1], Flexible Job Shop[1] a Flexible Flow Shop[1].

3.2.2 Charakteristika úloh

Charakteristiku úlohy specifikuje položka β. Jsou to omezení kladené na daný problém rozvrhování. Níže bude uvedeno několik základních a nejčastějších omezení.

Precedenční podmínky($prec$)

Tyto podmínky popisují vzájemné přednostní vztahy mezi jednotlivými úlohami. Tyto přednosti mohou být vyjádřeny například acyklickým orientovaným grafem. V praxi to může znamenat, že některá úloha j_2 nemůže začít dříve, než skončí úloha j_1.

Vhodnost stroje M_j

Vhodnost stroje má význam pouze v případě, že máme více paralelních strojů, které mohou provádět určitou úlohu j. Parametr M_j označuje množinu strojů způsobilých zpracovávat tuto úlohu j.

Omezení na pracovní sílu W

V případě, že do rozvrhování je ke strojům přidán další zdroj tj. operátoři, potom může být úloha na stroji vykonávána pouze v případě, že je v danou dobu k dispozici operátor k tomuto stroji.

Nastavování strojů a cena($sijk$, $cijk$)

Doba nastavování strojů značí potřebný čas, který nastává mezi dvěma úlohami z důvodu přenastavení stoje, na kterém jsou ihned po sobě tyto úlohy vykonávány. Cena za nastavování strojů nemusí být vždy podobná času přenastavení stroje. Stroj může být v některých případech přenastaven rychle, ale na druhou stranu cena za jeho přenastavení může být příliš vysoká. Může se jednat o situaci, kdy je na stroji vyměněn určitý náhradní díl za krátkou dobu, avšak nákupní cena tohoto náhradního dílu je obrovská.

[1]Pro tento název neexistuje v českém jazyce ekvivalentní překlad, proto je zmíněn v originálním znění. V dalších částech práce bude tento termín označován v originálním znění.

3.2.3 Optimalizační kritéria

Posledním parametrem pro rozvrhování je optimalizační kritérium. Názvy těchto kritérií jsou zavádějící. Například pro maximální dobu dokončení se jedná o rozvrh, kde se snažíme minimalizovat čas dokončení všech zakázek. Mezi nejpoužívanější optimalizační kritéria patří:

Maximální čas dokončení úloh (makespan)

Jestliže čas dokončení úlohy j značíme C_j, pak maximální dokončení všech úloh je $C_{max} = \max(C_1,...C_n)$.

Maximální nedochvilnost (lateness)

Nedochvilnost úlohy j značíme $L_j = C_j - d_j$, přičemž tato hodnota může být kladná i záporná. Maximální nedochvilnost všech úloh je $L_{max} = \max(L_1,...L_n)$.

Maximální opoždění (tardiness)

Opoždění úlohy j může nabývat pouze kladné hodnoty dle zápisu $T_j = \max(L_j,0)$

Celková vážená doba dokončení všech úloh

Vážená doba dokončení všech úloh je vyjádřena: $\Sigma\ w_j\ C_j$. Toto kritérium se používá při aplikaci prioritních zakázek do optimalizace.

Celková vážená doba trvání zakázek

Vážená doba trvání zakázek udává $\Sigma w_j(C_j - r_j)$.

Počet opožděných zakázek

Celkový počet opožděných zakázek vyjadřuje zápis:

$$\Sigma\ U_j = \begin{cases} 1 \text{ když } C_j > d_j \\ 0 \text{ v ostatních případech} \end{cases}$$

V mnoha situacích komplexní systémy vyžadují optimalizaci podle více než jednoho kritéria. Bohužel většina příkladů v literaturách se zabývá řešením pouze jednoho optimalizačního kritéria. Optimalizace podle více než jednoho kritéria by mohlo být dosaženo například váženým součtem dvou čí více kritérií, přičemž by bylo cílem tento součet minimalizovat. V této situaci však nastává problém, že může dojít ke spojení dvou či více kritérii, které jdou proti sobě. Například minimalizace makespan jde v zásadě proti účelové funkci opoždění zakázek.

3.3 Multi-operační problémy

Jak již bylo vysvětleno v kapitole 3.2.1, multi-operační (shop) problémy jsou typické tím, že se skládají z množiny úloh a množiny strojů. Jedna úloha se skládá z množiny operací, které musí být vykonány v daném pořadí na daných strojích a často mezi nimi bývají různé precedenční vztahy. Existuje několik typů těchto Shop problémů. Každý typ je něčím specifický a odlišný od ostatních. Shop problémy jsou klasické a velice známé optimalizační problémy z oblasti rozvrhování. Řadí se do kategorie NP-úplných problémů, jelikož se jedná o jeden z nejobtížnějších problémů kombinatorické optimalizace. Cílem rozvrhování je, aby každá operace byla zpracována na předem určeném stroji, přičemž vykonání této operace zabere určitý čas. Jinak řečeno Shop problém tedy spočívá v nalezení plánu, který alokuje čas strojů pro dané operace a splňuje následující podmínky:

- Zpracování jedné operace nemůže přerušit jiná operace.
- Každý stroj může vykonávat nejvýše jednu operaci najednou.

Existuje několik typů Shop problému - Job Shop, Flexible Job Shop, Flow Shop, Flexible Flow Shop, Open Shop. Tyto Shop problémy bývají široce používány při rozvrhování v průmyslové výrobě a jsou speciálními případy obecného Shop problému. Obecný Shop problém je definován následujícím způsobem. Máme n úloh $i = 1, ..., n$ a m strojů $m_1, ...m_m$. Každá úloha se skládá z množiny operací O_{ij} s určitou dobou vykonávání. Každá operace musí být vykonávána na nějakém stroji $m_1, ..., m_m$. Každá úloha může být vykonávána pouze na jednom stroji současně a každý stroj může najednou zpracovávat pouze jednu úlohu.

Job Shop

Job shop problém je zvláštním případem obecného shop problému, který zobecňuje flow shop problém. Máme n úloh $i = 1, ..., n$ a m strojů $M_1, ..., M_m$. Úloha i se skládá ze sledu n_i operací $O_{i1}, O_{i2}, ..., O_{in}$, které musí být prováděny v tomto pořadí. Z tohoto pořadí vyplývají precedenční omezení formulované $O_{ij} \rightarrow O_{ij+1} (j = 1, ..., n_i - 1)$. Každá operace O_{ij} je vykonávána na určitém stroji u_{ij} z množiny $\{M_1, ..., M_m\}$ s dobou trvání této operace p_{ij}.

Klasická verze job shop problému je adekvátní výrobním linkám v kusové výrobě, kde je větší počet odlišných výrobků. Tento typ problému je tedy spojován s kusovou výrobou, která probíhá formou výrobních zakázek. V tomto případě může mít každá úloha libovolný počet operací.

Flexible Job Shop

Problém flexible job shop je rozšířeným modelem job shop problému. V tomto problému se jedná o skutečnost, že operace O_{ij} může být vykonávána na kterémkoliv stroji z množiny Mij, kde musí platit $Mij \subseteq M$. Doba zpracování operace O_{ij} na vybraném stroji u z množiny M_{ij} je p^u_{ij}. Ve většině úloh je cílem vybrat pro každou operaci O_{ij} způsobilý stroj a počáteční čas takový, aby maximální doba dokončení všech úloh C_{max} byla co nejvíce minimalizována.

Flow Shop

Flow shop problém je speciální problém obecného shop problému kde:

- Každá úloha i se skládá z m operací O_{ij} s dobou zpracování $p_{ij}(j = 1, ..., m)$, kde O_{ij} musí být zpracovány na stroji M_j.
- Existují zde precedenční omezení $O_{ij} \rightarrow O_{ij+1}$ $(i = 1, ..., m-1)$. Každá zakázka je nejprve vykonávána na stroji 1, poté stroji 2, stoji 3 atd.

Mezi základní omezení a pravidla patří:

- Každá úloha se skládá z m operací, které musí být vykonány na m strojích.
- Každý stroj může vykonávat nejvýše jednu operaci jedné úlohy současně.
- Všechny úlohy provádí operace na strojích v pevně daném - stejném pořadí.
- Zpracování jedné operace nemůže přerušit jiná operace.
- Mezi operacemi rozdílných úloh není žádné precedenční omezení.

Cílem rozvrhování tohoto problému je nalézt pořadí operací, které bude minimalizovat čas dokončení všech úloh. Tento typ úlohy je adekvátní výrobě, kde jsou výrobní linky v sériové výrobě. V důsledku to znamená, že každá úloha musí být prováděna na všech strojích ve stejném pořadí. Pokud je stanoveno pořadí úloh na prvním stroji, pak je totožné pořadí použito také u všech ostatních strojů. Jedná se například o pásovou výrobu piva, kde musí být vždy dodrženo pevně stanovené pořadí operací.

Flexible Flow Shop

Je pouhým zobecněním typu Flow Shop problému. Tento problém se od flow shop problému liší v tom, že v každém stupni výroby se nenachází pouze jeden možný stroj M_j pro danou operace O_{ij}, ale existuje množina takových paralelních stojů, na kterých lze tuto operaci vykonávat. V každém stupni výroby je pro danou úlohu vybrán jeden stroj z množiny paralelních možných strojů.

Open Shop

Již z překladu lze usoudit, že se bude jednat o otevřený - volný režim rozvrhování. Tento typ je podobný Flow Shop problému, avšak s takovou odlišností, že pořadí provádění operací je libovolné. Tento typ rozvrhování lze přirovnat k plánování školního rozvrhu, kde v klasickém případě nezáleží na tom, v jakém pořadí se předměty budou vyučovat.

- Každá úloha i se skládá z m operací $O_{ij}(j = 1, ..., m)$, kde O_{ij} musí být zpracovány na stroji M_j.
- Nejsou zde žádné precedenční vztahy mezi jednotlivými operacemi.

3.4 Typy rozvrhů

Rozvrhy můžeme rozdělit na tři typy, které jsou znázorněny na obrázku 3.

Semi-aktivní rozvrh

Semi-aktivní rozvrh je takový rozvrh, kde není možné naplánovat žádnou operaci dříve, aniž by došlo ke změně pořadí, ve kterém vstupuje operace na daný stroj.
Mějme situaci, kde potřebujeme rozvrhnout dvě zakázky na tři stroje:

- Doba zpracování první zakázky na strojích č. 1 a 2 je rovná 1 hodině.
- Zakázka č. 2 je zpracována na strojích 2 a 3 shodně 2 hodiny.

Předpokládejme, že máme rozvrh, kde je zakázka č. 2 zpracovávána na stroji č. 2 před zakázkou č. 1. Tento rozvrh je semi-aktivní, protože nelze naplánovat žádnou operaci dříve, aniž by došlo ke změně pořadí operací na daném stroji. Nicméně tento rozvrh není aktivním rozvrhem z důvodu, že zakázka č. 1 může být na stroji č. 2 zpracovávána před zakázkou č. 2, aniž by došlo ke zpoždění této zakázky č. 2 na tomto stroji. Tento případ semi-aktivního rozvrhu lze vidět na obrázku vlevo 2.

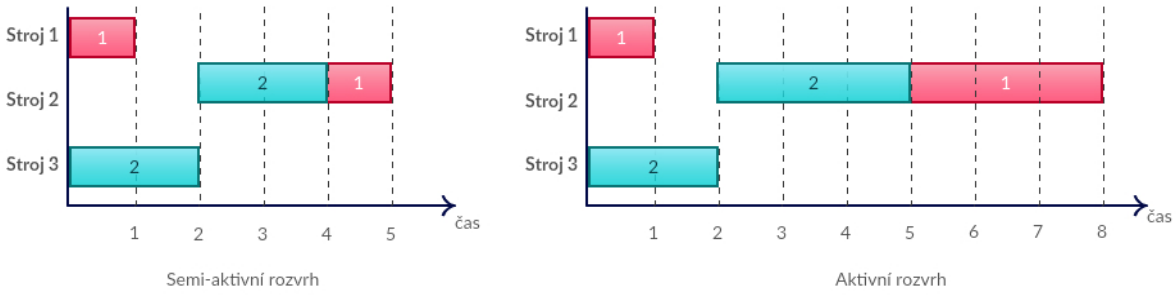
Aktivní rozvrh

Aktivní rozvrh je takový rozvrh, ze kterého není možné vytvořit jiný rozvrh změnou sledu alespoň jedné operace na stroji takovým způsobem, aby byla odvedena dříve a zároveň, aby se jiná operace tímto opozdila. Z obrázku 3 je zřejmé, že aktivní rozvrh musí být také semi-aktivním rozvrhem.

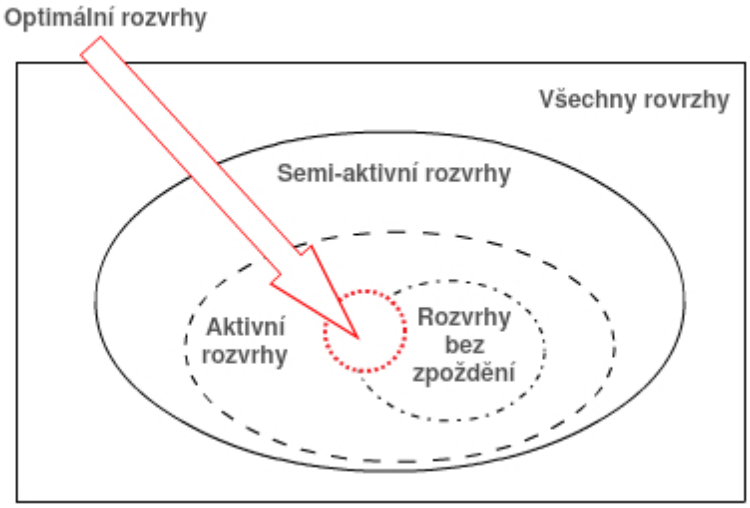
Můžeme si představit situaci, kde máme dvě zakázky a tři stroje:

- Zakázka č. 1 potřebuje 1 hodinu na stroji č. 1 a 3 hodiny na stroji č. 2.
- Zakázka č. 2 potřebuje 2 hodiny na stroji č. 3 a 3 hodiny na stroji č. 2.
- Obě zakázky musí být dokončeny na stroji č. 2, to znamená, na stroji č. 2 bude provedena jejich poslední-druhá operace.

Mějme rozvrh,ve kterém na stroji č. 2 je nejdříve zpracovávána zakázka č. 2 a až poté zakázka číslo 1. Tento rozvrh je aktivní, protože pokud by došlo k prohození operací na stroji č. 2, zapříčinilo by to u zakázky č. 2 opoždění oproti původnímu plánu. Každopádně tento plán není plánem bez zpoždění, jelikož stroj č. 2 zůstává první dvě hodiny v nečinnosti, přestože je po první hodině již k dispozici zpracování zakázky č. 1. Vpravo na obrázku 2 je tento rozvrh znázorněn.



Obrázek 2: Ukázka rozvrhů



Obrázek 3: Typy rozvrhů

Rozvrh bez zpoždění

Tento typ rozvrhu je v praxi nepoužívanější. Je to typ rozvrhu, kde stroj není bez práce, pokud je jakákoliv operace v daný čas pro tento stroj dostupná. Rozvrh bez zpoždění musí být také aktivním rozvrhem, avšak toto pravidlo nemusí platit opačně.

3.5 Složitost rozvrhování

Problémy s kvalitními řešeními (optimalizovanými rozvrhy) jsou spojeny s tím, že job shop problémy a většina dalších problémů s rozvrhováním strojů jsou NP-úplné problémy. Efektivně řešitelné jsou pouze některé konkrétní případy:

- Job shop problém s dvěma stroji, kde se zároveň úlohy můžou skládat z maximálně dvou operací.

- Job shop problém s dvěma stroji, ve kterých úloha je jedna samotná operace a je vykonána v určité jednotce času.

Úlohy problému rozvrhování patří téměř vždy do třídy NP, což znamená, že tyto problémy jsou nedeterministicky polynomiální. Vzhledem k tomu, že existuje $n!$ kombinací přiřazení n operací pro jeden stroj a strojů máme m v job shop problému, potom existuje $(n!)^m$ možných řešení. I pokud by byly brány v úvahu pouze možné plány, to znamená ty, kde je dodrženo veškeré omezení, tak ve srovnání s počtem všech kombinací je to stále neproveditelné.[10]

4 Řešení problematiky kapacitního plánování

V následující kapitole jsou představeny nejčastější metody řešení problému s rozvrhováním. Všechny tyto metody lze popsat pouze v obecném měřítku, jelikož jejich aplikace se téměř v každé implementaci velice liší. Tato odlišnost je dána přesnějším řešením konkrétního problému s různými specifiky. V oblasti rozvrhování je prohledávaný prostor možného optimálního řešení obrovský a je klasifikován jako NP-úplný problém. Řešení tohoto problému lze nalézt pomocí přesných (exaktních) metod použitelných do určitého rozsahu problému. Při větším rozsahu problému jsou tyto metody neefektivní. Dalšími typy algoritmů jsou algoritmy heuristické či evoluční.

4.1 Přesné metody

Mezi nejznámější přesné metody, které jsou využívány pro problémy s rozvrhováním je metoda větví a mezí či metoda dynamického programování. Faktem je, že metoda větví a mezí i metoda dynamického programování jsou na jednu stranu velice účinné metody, ale na druhou stranu jejich časové složitosti jsou často příliš vysoké a nepřijatelné při řešení NP-úplných problémů. [7]

4.1.1 Metoda větví a mezí

Metoda větví a mezí je nejčastěji založena na reprezentaci výsledného rozvrhu pomocí disjunktivního grafu. Samotný algoritmus využívá stromové struktury prostorového řešení daného problému. Tento prostor je postupně dělen na menší prostory. Špatné prostory z optimalizačního hlediska jsou z prohledávání vylučovány. Tato metoda se řadí mezi algoritmy přesné a hledá optimální řešení, které závisí na proceduře větvení a horním či dolním odhadu mezí. [4]

Algoritmus se skládá ze dvou důležitých operací, kterými jsou větvení a výpočty mezí. Větvení je operace, která rozdělí množinu na dvě či více disjunktních podmnožin. Algoritmus začíná na uzlu nejvyšší úrovně a končí, jakmile je vyhodnocen uzel na nejnižší úrovni. Každý uzel na určité úrovní u ve stromové struktuře reprezentuje částečnou sekvenci u operací. Dosud nevybrané uzly na hladině u určují další potencionální možné větve, které budou následovat pod úrovní u. Takto se dále opakovaně větví i tyto podmnožiny, jejich podmnožiny atd. Při každém větvení se pro podmnožinu určí dolní mez pro minimalizaci a horní mez při maximalizaci. U rozvrhování se bude jednat logicky o minimalizaci, protože je žádoucí, aby byl co nejmenší čas výroby, popřípadě minimální využití materiálů či jiných zdrojů. Pro další větvení se následně volí ta podmnožina, která má nejnižší dolní mez. V případě, že dolní mez pro některý uzel stromu je větší, než je horní mez některého jiného uzlu, pak uzel stromu může být bezpečně odstraněn z vyhledávání. Proces větvení a výpočtu mezí se provádí, dokud není dosaženo kompletního rozvrhu pro každý stroj. Cílem tohoto algoritmu je najít řešení, jehož hodnota není vyšší, než je kterákoli dolní mez.

Při řešení úloh s rozvrhováním touto metodou jsou jednotlivé uzly reprezentovány jako neúplné rozvrhy. Tyto uzly jsou postupně následným větvením konstruovány tak, aby nebylo porušeno precedenčních podmínek. Takovou podmínkou může být pořadí operací v dané zakázce. Tato problematika rozvrhování pomocí metody větví a mezí se řeší v literatuře [4].

Algorithm 1 Pseudokód algoritmu větví a mezí pro Job Shop problém podle [11]

```
1:  function Branch-and-Bound(r)
2:      Vytvořit řešení S ∈ Y(r) použitím heuristiky
3:      if C_max(S) < UB then
4:          UB ← C_max(S)
5:      end if
6:      Vypočítat kritickou cestu P
7:      Vypočítat bloky P
8:      Vypočítat množiny E_j^B a E_j^A
9:      while operace i ∈ E_j^v existuje u j = 1, ..., k a zároveň v = A, B do
10:         Vymazat i z E_j^v
11:         Opravit disjunkce pro odpovídajícího následníka s
12:         Vypočítat spodní hranici LB(s) pro uzel s
13:         if LB(s) < UB then
14:             Branch-and-Bound(s)
15:         end if
16:     end while
17: end function
```

4.2 Heuristické metody

Heuristické algoritmy jsou metody, které nezaručují nalezení nejlepšího/optimálního řešení. Tyto algoritmy se používají zejména v situacích, kdy selhávají exaktní algoritmy, které by nenašly řešení v rozumném čase. Jinak řečeno, tyto algoritmy se používají tam, kde zatím není známý lepší exaktní algoritmus pro nalezení přesného optimálního řešení. Typickou úlohou, která se nejčastěji řeší pomocí heuristických metod je NP-úplná úloha - problém obchodního cestujícího. V této úloze jde o nalezení nejkratší cesty na mapě. Postupně se tyto metody rozšířily jako řešení v rámci optimalizace rozvrhování a plánování.

Heuristické algoritmy můžeme rozdělit na deterministické a stochastické. Stochastické algoritmy jsou specifické v tom, že některé jejich kroky využívají náhodných operací. To ve výsledku znamená, že jimi získané výsledky řešení se v jednotlivých bězích programu mohou lišit. Není proto od věci, spustit tento algoritmus vícekrát a poté vybrat nejlepší získané řešení. Stochastické heuristické metody se někdy také označují jako meta-heuristiky, protože poskytují pouze obecný rámec. V těchto algoritmech je potřeba zvolit vlastní operace algoritmu v závislosti na daném problému, který by algoritmus měl vyřešit. [8] Deterministické heuristické algoritmy mají tu vlastnosti, že vždy dojdou ke stejnému výsledku stejnou cestou.

Typickými zástupci heuristických metod jsou: genetické algoritmy (GA), simulované žíhání (SA), zakázané hledání (TS), evoluční algoritmy, lokální hledání a neuronové sítě. Často jsou tyto algoritmy navrženy pro řešení specifického typu úlohy.

4.2.1 Konstruktivní algoritmy

Konstruktivní algoritmy jsou zpravidla jednoduché, zřetelné a tudíž jsou jednoduché na pochopení i implementaci. Jedná se o nejčastěji používané algoritmy pro řešení problémů rozvrhování v praxi. Tyto algoritmy mají při svém startu prázdné řešení (rozvrh) a postupnou konstrukcí pomocí prioritních pravidel či rozvrhování dle úzkého místa je rozvrh vytvářen.[2] Tento algoritmus zároveň zaručuje takovou konstrukci rozvrhu, aby splnil všechna precedenční či zdrojová omezení.

Prioritní pravidla

Prioritní pravidla jsou nejpoužívanější heuristiky pro řešení problémů rozvrhování v praxi z důvodu jejich nízké časové složitosti a snadné implementaci. Tento algoritmus postupuje způsobem postupného přidávání operací do rozvrhu z aktuálně dostupných operací podle určitého pravidla. První dostupnou operaci lze nazvat takovou, která zatím není umístěna v rozvrhu a je jako první na řadě v technologickém postupu. Z takto dostupných operací se vybírá stroj, na kterém by tato operace mohla začít nejdříve. U takového stroje pak je nutné rozhodnout, které z dostupných operací dát přednost. Toto rozhodování určují právě prioritní pravidla. Takových prioritních pravidel bylo v 70. letech minulého století známo více než sto. [2] Nejznámější a nepoužívanější prioritní pravidla zachycuje tabulka 3.

Mějme n úloh, zadané procesní časy p_j, čas připravenosti $r_j = 0$ a m paralelních strojů. Cílem je najít minimální délku rozvrhu C_{max}, pomocí algoritmu prioritního pravidla - nejdelší doba zpracování. Pseudokód je uvedení níže.

Algorithm 2 Pseudokód algoritmu s prioritním pravidlem - nejdelší doba zpracování

Vytvoř seznam úkolů setříděný podle sestupně podle procesních časů
1: **for** $j \leftarrow 1$ to m **do**
2: $t_j \leftarrow 0$ (Stroje jsou na začátku volné)
3: **end for**
4: $j \leftarrow 1$
5: **repeat**
Vybrat stroj s nejkratším rozvrhem, $t_k = min\{t_j\}$
Přiřaď úlohu T_j stroji P_k v čase t_k
6: $s_k \leftarrow s_k + p_j$
7: $j \leftarrow j + 1$
8: **until** j = n

	Prioritní pravidlo	**Popis**
1.	Nejkratší doba operace	Operace s nejkratší dobou zpracování na zvažovaném stroji.
2.	Nejdelší doba operace	Operace s nejdelší dobou zpracovaní na zvažovaném stroji.
3.	Nejdelší zbývající doba zpracování	Operace s nejdelší zbývající dobou zpracování zakázky
4.	Nejkratší zbývající doba zpracování	Operace s nejkratší zbývající dobou zpracování zakázky
5.	Nejdelší operace zbývajícího času zpracování	Operace s nejvyšší sumou ukončení a dobrou zpracování
6.	Náhodně	Operace pro zvažovaný stroj je vybrána náhodně
7.	Kdo dříve přijde, ten jde dříve.	První operace ve frontě zakázek čekající na stejný stroj.
8.	Nejkratší doba zpracování	Zakázka s nejkratší celkovou dobou zpracování.
9.	Nejdelší doba zpracování	Zakázka s nejdelší celkovou dobou zpracování.
10.	Nejdelší následující operace	Operace s nejdelší dobou zpracování následující operace.
11.	Nejmenší počet zbývajících operací	Operace s nejmenším počtem následujících operací na zakázce.
12.	Největší počet zbývajících operací	Operace s největším počtem následujících operací na zakázce.

Tabulka 3: Nejpoužívanější prioritní pravidla podle [9]

Výhodou tohoto způsobu rozvrhování je rychlé a jednoduché generování výrobních rozvrhů. Hlavní nevýhodou však je, že nelze zaručit nalezení optimálního rozvrhu a možnost dosáhnout stejného řešení v případě generování pomocí rozdílných prioritních pravidel. [2] Použití této metody je ideální v případě, kde je potřeba rychlého generování rozvrhu a není potřeba příliš vysoké optimalizace rozvrhu. Dalším případem použití této metody může být generování vstupních řešení pro prohledávací heuristické metody, jako jsou například lokální hledání či zakázané hledání.

Posouvání úzkého místa

Posouvání úzkého místa je technika, která patří mezi úspěšné a populární heuristiky pro job shop problémy. Poprvé byla tato metoda zveřejněna v 80.letech minulého století a dnes existuje hned několik variant této metody. Tento algoritmus postupuje iterativně, kde v každé iteraci je identifikován problémový stroj v rozvrhu (úzké místo). Jakmile algoritmus najde stroj, který je pro daný rozvrh úzkým místem, tak vybere optimální sekvenci operací pro tento stroj a podle této sekvence se musí přeuspořádat všechny ostatní již rozvržené stroje. [10] Celý tento postup se opakuje, dokud nejsou rozvrženy všechny stroje. Nevýhodou tohoto algoritmu je, že negarantuje nalezení optimálního řešení.

4.2.2 Lokální hledání

Metody založené na lokálním prohledávání patří mezi nejstarší metody založené na prohledávání prostoru řešení. Tyto metody patří mezi nejjednodušší a nejuniverzálnější metody. Nevýhodou těchto algoritmů je časté uvíznutí v lokálním extrému, což způsobuje vygenerované počáteční řešení. Ukončení těchto algoritmů obvykle závisí na počtu iterací či ukončovací podmínce založené na uvíznutí v lokálním extrému.[2] Mezi nejpoužívanější metody lokálního hledání patří metoda simulovaného žíhání (SA) a metoda zakázaného hledání(TS).

Simulované žíhání

Algoritmus simulovaného žíhání je optimalizační metoda prohledávající stavový prostor, který je založený na principu žíhání oceli. Samotný název algoritmu je přímo odvozen podle fyzikálního procesu ochlazování kovu, díky němuž se odstraňují defekty v krystalové mřížce kovu. Takový kov se nejdříve zahřeje na určitou vysokou teplotu a poté se postupně ochlazuje (žíhá). Hlavním parametrem tohoto algoritmu je teplota, na níž závisí velikost změny daného řešení v průběhu algoritmu. Čím je tato teplota vyšší, tím větší změny v průběžném pořadí se provádí. V průběhu algoritmu jsou s určitou pravděpodobností přijímána i horší řešení, než je aktuálně nejlepší řešení. Teplota je postupně snižována v závislosti, jak rychle se přibližujeme k cíli (optimálnímu rozvrhu).

Pokud je potřeba rozvrhnout zakázky tak, aby byly co nejdříve hotové, jedná se tedy o minimalizační optimalizační problém C_{max}. V tomto případě bereme krystal, jako řešení daného problému (rozvrh). Každý tento krystal má svou určitou energii, což je funkční hodnota účelové funkce. V tomto případě se jedná o celkový výrobní čas daného rozvrhu. Pokud máme průběžné řešení rozvrh R, pak se postupuje náhodným výběrem sousedního řešení R' z okolí $U(R)$. Pokud je sousední řešení R' lepší než původní řešení R, automaticky dále pokračuje sousední řešení R'. Pokud je však sousední řešení horším řešením, neznamená to, že bude automaticky zahozeno. S určitou pravděpodobností může v algoritmu dále pokračovat dle Metropolisova vzorce 2, kde $f(R')$ a $f(R)$ jsou hodnoty účelové funkce a T je aktuální teplota. Čím nižší je teplota, tím menší je pravděpodobnost přijetí horšího řešení, než je současné nejlepší řešení.

$$P = e^{(\frac{f(R')-f(R)}{T})} \tag{2}$$

Teplota T má stavenou horní a dolní hranici, takže platí $T_{min} \leq T \leq T_{max}$. Obvykle se teplota T postupně snižuje pomocí vzorce 3, kde α se pohybuje obvykle v rozsahu 0,8 - 0,99.

$$T = \alpha * T \tag{3}$$

Po celou dobu algoritmu nemusí být pokles teploty T stejný. Parametry pro snižování teploty, či počet opakování prohledávání pro určitou teplotu lze měnit za běhu algoritmu v závislosti na tom, jak rychle algoritmus směřuje k optimálnímu řešení či naopak.

Algorithm 3 Pseudokód algoritmu simulovaného žíhání, převzato z [8]

$x_0 \leftarrow$ Náhodný vygenerované řešení
$x_{best} \leftarrow x_0$
$T \leftarrow T_{max}$
$k \leftarrow 1$
while $T > T_{min}$ **do**
    for $k \leftarrow 1$ to max. počet opakování pro danou teplotu **do**
        Vyber x z množiny sousedů $N(x_0)$
        $\delta f \leftarrow f(x) - f(x_0)$
        if $\delta f < 0$ **then**
            $x_0 \leftarrow x$
            if $f(x) < f(x_{best})$ **then**
                $x_{best} \leftarrow x$
            end if
        else
            Náhodně vygeneruj číslo r z intervalu (0,1)
            if $r < e^{(\frac{f(R')-f(R)}{T})}$ **then**
                $x_0 \leftarrow x$
            end if
        end if
    end for
    $T \leftarrow \alpha \cdot T$
end while

4.3 Genetické a evoluční algoritmy

Poprvé byly genetické algoritmy představeny Johnem Hollandem (1975). Jsou založeny na principech evoluce, používané v přírodě a popsané Charlesem Darwinem. Většina pojmů jako jsou populace, jedinec, generace byla převzata přímo z biologie. Typickým rysem klasických genetických algoritmů je, že jedinci jsou zakódováni v binární soustavě. [8] Tyto algoritmy využívají evolučních procesů jako jsou dědičnost, mutace, selekce a křížení pro optimalizaci daného řešení. Dalšími termíny, se kterými se v těchto algoritmech setkáváme jsou populace a jedinec. Populace obsahuje množinu jedinců, kteří by se měli v průběhu řešení zlepšovat. Jedincem se rozumí jedno konkrétní řešení daného problému. V každé generaci je potřeba u všech jedinců v dané populaci vypočítat fitness hodnotu, která udává kvalitu řešení danou jedincem. Na základě těchto fitness hodnot, jsou tito jedinci buďto z populace vyřazeni, nahrazováni lepšími jedinci, mutováni nebo kříženi s jinými jedinci. Tyto operace jsou vždy typické pro konkrétní algoritmus. Algoritmus je ukončen pokud je splněna ukončovací podmínka, což obvykle bývá zadaný počet generací. Obecný algoritmus je vyjádřen pseudokódem 4.

Evoluční algoritmy přísluší do třídy stochastických prohledávacích algoritmů a pracují s náhodnými změnami navrhovaných řešení.

Algorithm 4 Pseudokód evolučních/genetických algoritmů

```
1: Vygeneruj počáteční populaci G
2: i ← 0
3: repeat
4:     Ohodnocení každého jedince v populaci dle účelové funkce
5:     Evoluční proces - evoluční operátory daného algoritmu,výběr/vytvoření nových potomků
   do další populace
6:     i ← i+1
7: until Dokud není splněna podmínka ukončení
```

4.4 Algoritmy s prvky umělé inteligence

Existují také algoritmy, které lze zařadit do kategorie algoritmů umělé inteligence, jelikož jsou tyto algoritmy aplikovány podle chování organizovaných systémů v přírodě. Mezi nepoužívanější a neznámější algoritmy patří algoritmus mravenčích kolonií (ACO) a algoritmus rojení částic (PSO).

Algoritmus PSO, který vznikl v roce 1995, je inspirován sociálním chováním ptačích a rybích hejn. V tomto algoritmu je každá částice definovaná svou polohou, rychlostí a pamětí předchozích úspěchů při hledání řešení. Nejúspěšnější částice ovlivňují méně úspěšné částice z hejna.

Algoritmus ACO je technika vyvinutá v roce 1991, která se inspiruje chováním mravenců při hledání potravy. Algoritmus využívá při výpočtu množinu „mravenců"“ kteří jsou tzv. agenti. Tito agenti spolu komunikují pomocí feromonové stopy na cestě k nalezené potravě. Tímto způsobem kolonie mravenců nalézá nejkratší cestu k potravě. Čím delší cesta za potravou je,

tím rychleji feromonová stopa vyprchává. Mravenci se s určitou pravděpodobností rozhodují pro tu cestu, která má silnější feromonovou stopu.

5 Návrh a implementace modulu pro kapacitní plánování

Tato kapitola se zabývá představením daného byznys problému, pro který bude modul kapacitního problému navržen. Dále následuje výběr algoritmu, návrh datových struktur pro tento algoritmus a nastínění implementace tohoto modulu.

5.1 Řešený byznys problém

Modul pro kapacitní plánování bude testován na datech nejmenované výrobní firmy. Hlavními prvky této výroby, které je potřeba brát při rozvrhování operací v potaz jsou:

- Výrobní postup
- Dostupné výrobní stroje
- Zakázky
- Operace

Výrobní postup je definice jak vyrobit určitý výrobek. Typicky to bývá sled po sobě jdoucích operací s určitými parametry. Dostupnými výrobními stroji se chápou stroje, které můžeme použít pro výrobu určitého výrobku či polotovaru. Výrobní stroje jsou uspořádány do určitých skupin, které nazýváme strojní centra. Aby bylo co vyrábět, je potřeba mít poptávku, což zaručují zakázky. Zakázka udává, jaký výrobek vyrobit a v jakém množství. Pro vyrobení určitého výrobku nepotřebuje pouze stroje, ale zakázku je potřeba rozdělit do několika elementárních kroků, které se nazývají operace.

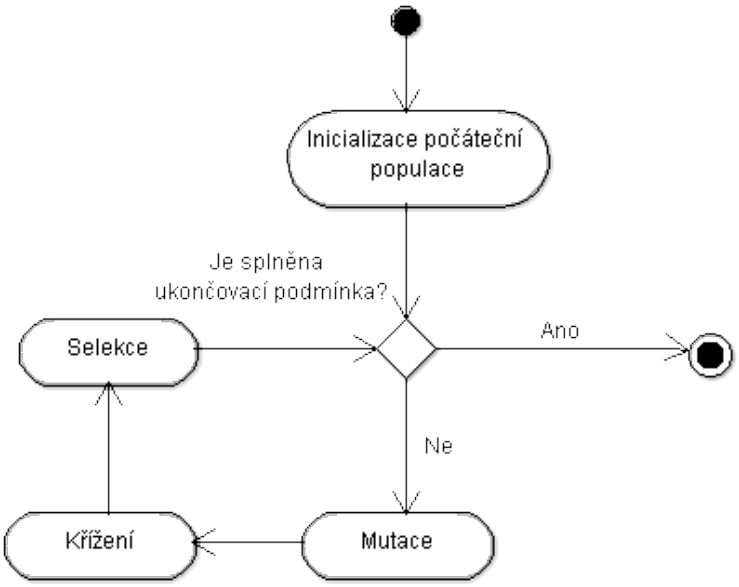
Pro operaci bývá definováno, na kterém strojním centru může být vykonávána. Jelikož ve většině případů má strojní centrum více než jeden stroj, není přesně určen konkrétní stroj, kde může být operace vykonávána. Cílem při rozvrhování je tedy určit nejen čas a pořadí, jak se budou operace vykonávat, ale taky určit přesný stroj.

Rozvrhování výroby bude aplikováno na všechny stroje ve výrobě. Zakázky se obvykle skládají z 5-10 po sobě jdoucích operací. Ve výše popsaném byznysu se jedná o zakázkovou výrobu a dle těchto informací je identifikovatelné, že se jedná o flexible job shop problém.

Jelikož se byznys procesy v různých výrobních podnicích liší, nelze vytvořit jeden univerzální algoritmus pro rozvrhování, který by pasoval na všechny typy výroby. Tento modul může být v budoucnu rozšiřován o řešení dalších problémů kromě flexible job shop problému. To však není cílem této práce. V této práci se tedy modul kapacitního plánování věnuje řešení flexible job shop problému pro zmiňovaný byznys problém.

5.2 Diferenciální evoluce

Diferenciální evoluce (DE) je evoluční algoritmus, který v roce 1995 vyvinuli a poprvé použili Ken Price a Rainer Storm. Experimentální výsledky ukazují, že často konverguje rychleji než jiné



Obrázek 4: Typický průběh algoritmu diferenciální evoluce

stochastické algoritmy pro globální optimalizaci. Samotný průběh diferenciální evoluce vystihuje obrázek 4.

5.2.1 Parametry algoritmu

Účinnost diferenciální evoluce ovlivňují vstupní řídící parametry jako u jiných evolučních algoritmů. Těmito parametry jsou:

- Velikost populace
- Mutační konstanta F
- Pravděpodobnost křížení Cr
- Maximální počet generací
- Dimenze D

Velikost populace udává počet jedinců v populacích. Přesná velikost tohoto parametru není přesně stanovena, a je získána pouze zkušeností s tímto algoritmem. Je však dokázáno, že minimální velikost populace nesmí být menší než 4. Obvykle doporučovaná hodnota je v intervalu $[10D,100D]$, kde D je velikost dimenze řešeného problému neboli počet parametrů účelové funkce.[8]

Mutační konstanta F je parametr, jejíž hodnota hraje roli při vytváření nového parametru uvnitř jedince.

Křížení Cr je konstanta, která řídí proces křížení. Pokud se jedná o funkci, která je separabilní, je nutno nastavit parametru hodnotu blízkou nule. V opačném případě je výhodnější nastavit hodnoty blížící se 1. Při hodnotě 0 bude docházet k tomu, že se mutace nedostane do zkušebního jedince a tento zkušební jedinec bude pouze čistou kopií nejlepšího jedince. Pokud bude hodnota $Cr = 1$, pak se bude algoritmus podobat spíše náhodnému hledání než evolučnímu algoritmu.[8]

Maximální počet generací je parametr, který udává délku algoritmu a počet evolucí. Tento parametr je naprosto volitelný a lze ho měnit v průběžně v závislosti na tom, jestli se jedinci evolucí stále vyvíjí, nebo zda došlo ke stagnaci.

5.2.2 Inicializace

Před samotnou inicializací populace je důležité stanovit horní hranici (b_U) a dolní hranici (b_L) pro hodnoty parametrů ve vektorech. Jakmile jsou tyto hranice stanoveny, dochází k inicializaci parametrů jedince pomocí náhodného generování čísel. Tyto čísla jsou v rámci prohledávaného prostoru(určuje hodní a dolní hranice). V počáteční populaci G =0 je pro každý j-tý parametr i-tého vektoru vygenerována hodnota dle vzorce 4. Tento vzorec zajišťuje, že hodnota je vygenerována v rámci prohledávaného prostoru odpovídajícího dolní a horní hranicí.

$$x_{j,i,0} = rand_j(0,1) \cdot (b_{j,U} - b_{j,L}) + b_{j,L} \tag{4}$$

5.2.3 Mutace

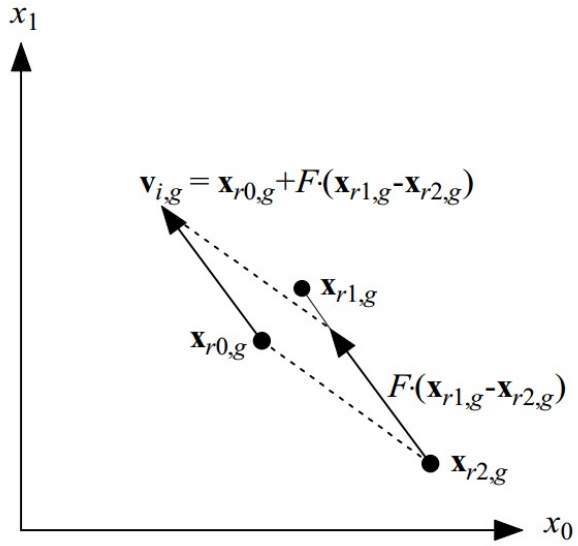
Jakmile je populace inicializována, DE algoritmus pokračuje mutací k vytvoření šumových vektorů. Vzorec 5 znázorňuje, že k procesu mutace jsou potřeba tři další náhodně vybraní jedinci, pomocí kterých je vytvořen šumový vektor $v_{i,g}$. Výběr základního indexu vektoru r_0 může být stanoven různými cestami v závislosti na vybraném typu algoritmu mutace DE. Indexy r_1 a r_2 jsou vybírány vždy náhodně z celé populace. Průběh vzniku nového parametru mutací vystihuje obrázek 5 .

$$v_{i,g} = x_{r0,g} + F \cdot (x_{r1,g} - x_{r2,g}) \tag{5}$$

5.2.4 Křížení

Po procesu mutace algoritmus pokračuje fází křížení, ve které se vybírá hodnota parametru vzniklá z mutace nebo hodnota parametru z určitého jedince. Zásadní vliv, která hodnota bude do zkušebního jedince převzata, má náhodně vygenerované reálné číslo z intervalu <0;1> a velikost parametru Cr. Postup křížení je v zápise 6.

$$u_{j,i,g} = \begin{cases} v_{j,i,g} \text{ když } (rand_j(0,1) \leq Cr \text{ nebo } j = j_{rand}) \\ x_{j,i,g} \text{ v ostatních případech} \end{cases} \tag{6}$$



Obrázek 5: Mutace v DE, převzato z [12]

5.2.5 Selekce

Pokud máme zkušební vektor vzniklý procesem křížení, pak je potřeba zvolit, zda bude nový jedinec do další populace zastupován tímto zkušebním vektorem, nebo vektorem aktuálního jedince z populace. Pokud je hodnota účelové funkce (fitness) zkušebního jedince nižší nebo rovna hodnotě účelové funkce aktuálního jedince, pak je do další populace přidán zkušební jedinec. Dle zápisu 7 v opačném případě do další populace pokračuje aktuální jedinec.

$$\vec{X_{i,G+1}} = \begin{cases} u_{i,G} \text{ když } f(u_{i,g}) \leq f(x_{i,g}) \\ x_{i,g} \text{ v ostatních případech} \end{cases} \tag{7}$$

5.2.6 Varianty diferenciální evoluce

Jednotlivé varianty diferenciální evoluce se pouze liší v části mutace. Každá z variant využívá jiný způsob vytváření šumového vektoru.

- DE/best/1/exp: $v = x^G_{best,j} + F \cdot (x^G_{r2,j} - x^G_{r3,j})$
- DE/rand/1exp: $v = x^G_{1,j} + F \cdot (x^G_{r2,j} - x^G_{r3,j})$
- DE/rand-to-best/1/exp: $v = x^G_{i,j} + \lambda \cdot (x^G_{best,j} - x^G_{i,j}) + F \cdot (x^G_{r2,j} - x^G_{r3,j})$
- DE/best/2/exp: $v = x^G_{best,j} + F \cdot (x^G_{r1,j} + x^G_{r2,j} - x^G_{r3,j} - x^G_{r4,j})$
- DE/rand/2/exp: $v = x^G_{5,j} + F \cdot (x^G_{r1,j} + x^G_{r2,j} - x^G_{r3,j} - x^G_{r4,j})$

Identifikace operace	O11	O12	O13	O21	O22	O23	O24	O31	O32
Jedinečné ID operace	1	2	3	4	5	6	7	8	9

Obrázek 6: Přidělení operacím jedinečné ID

5.3 Použití diferenciální evoluce pro rozvrhování

Pro správnou funkčnost a použití diferenciální evoluce v plánování je vyžadováno hned několik správných konfigurací. Bez nich by buďto algoritmus byl nefunkční, nebo v případě špatné konfigurace by algoritmus mohl být velice neefektivní.

Jelikož se v algoritmu diferenciální evoluce využívají vektory a vektorové operace, tak k efektivní funkčnosti je velice klíčové, aby byla reprezentace vektoru vhodně sestavena. Tato reprezentace je inspirována podle článku [3].

Druhým klíčovým faktorem je model účelové funkce, která určuje směr, kam se bude evoluce ubírat.

5.3.1 Reprezentace vektorů

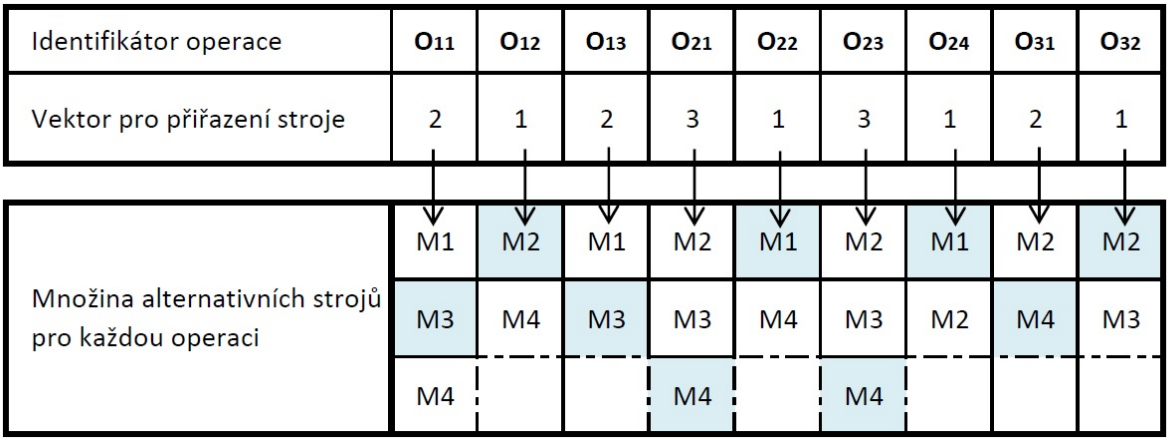
Při vytváření správného modelu vektoru je potřeba uvažovat, co má být výsledkem dané úlohy, a jaký zásadní problém je potřeba při evoluci řešit. Mějme vektor $\vec{X} = [x_1, x_2, ..., x_D]$, který reprezentuje dimenzi o velikosti D. V našem případě je $D = 2d$, kde d je počet všech operací, které se budou při rozvrhování řešit.

Vektor lze tedy rozdělit na dvě poloviny. Každá polovina má jiný význam pro výsledné řešení. První polovina vektoru $\vec{X_1} = [x_1, x_2, ..., x_d]$ značí, na jaký stroj bude přiřazena každá operace. Druhá polovina vektoru $\vec{X_2} = [x_{d+1}, x_{d+2}, ..., x_{2d}]$ udává informaci o tom, v jakém pořadí budou všechny operace vykonávány. Jelikož algoritmus pracuje s vektory v D dimenzionálním prostoru, je potřeba stanovit hranice daného prostoru. Pro každou hodnotu ve vektoru $\vec{X}$ je stavena hranice $[-\delta, \delta]$, kde $\delta > 0$. V této práci byla hranice nastavena konstantou $\delta = 20$. Každá hodnota ve vektoru $\vec{X}$ tedy může nabývat pouze hodnoty z intervalu [-20; 20]. Jako vstup algoritmu jsou všechny vektory v počáteční populaci náhodně inicializovány pomocí vzorce 4, přičemž splňují podmínku $x_{j,min} = -\delta$, $x_{j,max} = \delta$, $j = 1, 2, ..., D$.

Na obrázku 6 je znázorněno mapování všech operací do celého vektoru. Ve skutečnosti jsou postupně brány všechny operace ze zakázek a každé této operaci je přiřazené jedinečné ID, kde ID $\in N^0$.

Vektor pro přiřazení strojů

Vektor pro reprezentaci stroje k operacím je reprezentován $\vec{R} = [r_1, r_2, ..., r_d]$, kde každá z hodnot je celé číslo. Hodnoty r_j, kde $j = 1, 2, ..., d$ znamenají r_j-tý možný stroj z množiny alternativních strojů pro danou operaci, která odpovídá indexu j. Na obrázku 7 můžeme vidět příklad, kde vektor $\vec{R}$ udává přiřazení devíti operací na stroje, na kterých mohou být vykonávány. Pro příklad



Obrázek 7: Vektor pro přiřazení stroje

$r_6 = 3$ znamená, že operace O_{23} bude přiřazena na třetí stroj v pořadí z množiny alternativních strojů této operace. V tomto případě bude tato operace přiřazena na stroj číslo 4 (M4). Hodnota r_6 je mapována na operaci O_{23}, protože tato operace získala mapováním jedinečné ID = 6, jak bylo znázorněno dříve na obrázku 6.

Vektor pro určení sekvence operací

Vektor pro určení sekvence operací je reprezentován $\vec{S} = [s_1, s_2, ..., s_d]$ a je permutací ID všech operací. Pořadí každé operace ve vektoru $\vec{S}$ značí prioritu pro rozvrhování do rozvrhu. Pokud bude existovat vektor $\vec{S} = [1, 4, 8, 5, 2, 3, 6, 9, 7]$, potom to znamená, že operace budou rozvrhovány v následujícím pořadí: $O_{11} \rightarrow O_{21} \rightarrow O_{31} \rightarrow O_{22} \rightarrow O_{12} \rightarrow O_{13} \rightarrow O_{23} \rightarrow O_{32} \rightarrow O_{24}$. Operace O_{11} má největší prioritu a bude rozvrhována jako první. Poté bude následovat operace O_{21} atd.

Samozřejmě existují proveditelné i neproveditelné permutace ID operací. Může vzniknout permutace, která nesplňuje pořadí operací dané zakázky. Pokud bude brán v úvahu předchozí příklad, tak pak je zřejmé, že dle výrobní definice operace O_{12} nemůže být prováděna před operací O_{11}. V tomto případě je potřeba toto pořadí korektně opravit v rámci dané práce(zakázky).

5.3.2 Kódování a dekódování

Kódování rozvrhu probíhá konverzí daného rozvrhu do dvou vektorů $\vec{R}$ a $\vec{S}$. První vektor $\vec{R}$ získáme jednoduše právě přiřazeným strojům z rozvrhu, zatímco $\vec{S}$ je dán setříděním operací podle jejich startovacího času v rozvrhu.

Dekódování dvou vektorů na rozvrh se dělí do dvou části. V první části je potřeba dekódovat vektor $\vec{R}$, podle něhož ke každé operaci přiřadíme stroj, na kterém se má operace vykonávat. Před dekódováním druhého vektoru $\vec{R}$ je potřeba nejdříve provést korekci pořadí operací, aby splňovaly zadané pořadí v rámci zakázky. Poté je všem těmto operacím přiřazen nejlepší do-

stupný čas pro vykonávání na daném přiděleném stroji. Rozvrh vygenerovaný tímto způsobem je rozvrhem aktivním.

5.3.3 Dopředná konverze

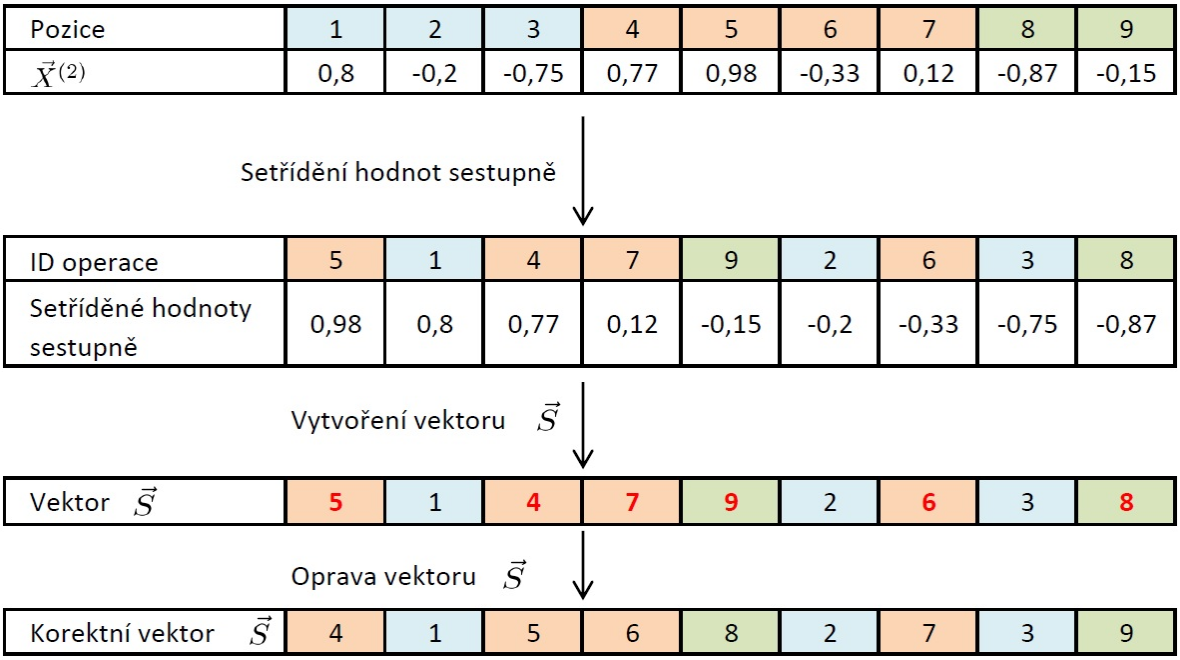
Dopřednou konverzí rozumíme konverzi, která nám z vektoru $\vec{X} = [x_1, x_2, ..., x_d, x_{d+1}, ...x_{2d}]$ vytvoří vektor pro přiřazení strojů $\vec{R}$ a vektor pro určení sekvenci operací $\vec{S}$. Jelikož chceme z jednoho vektoru vytvořit dva různé vektory, je potřeba tuto konverzi provádět ve dvou oddělených krocích.

V prvním kroku se vezme první polovina vektoru $\vec{X}^{(1)} = [x_1, x_2, ..., x_d]$ a provede konverze na vektor pro přiřazení strojů $\vec{R} = [r_1, r_2, ..., r_d]$. K tomu je potřeba ke každé operaci mít informaci, na kolika strojích může být vykonávána. K tomu slouží vektor $\vec{L} = [l_1, l_2, ..., l_d]$, kde každá hodnota uvnitř vektoru znamená počet alternativních(možných) strojů pro operaci s odpovídajícím ID, které odpovídá pozici této hodnoty ve vektoru. Všechny hodnoty l z vektoru $\vec{L}$ musí platit, že $l \geq 1$. Pokud vezmeme v úvahu příklad z obrázku 7, pak by vektoru alternativních strojů odpovídal vektor $\vec{L} = [3, 2, 2, 3, 2, 3, 2, 2, 2]$. Z toho vyplývá, že operace O_{23} s ID = 6 může být přiřazena na 3 různé stroje.
Důležitým krokem je správně převést reálné číslo $x_j \in [-\delta, \delta]$ na celé číslo $r_j \in [1, l_j]$. K tomuto převodu slouží vzorec 8, který pomocí lineární transformace převede reálné číslo x_j na reálné číslo v rozsahu $[1, l_j]$. Toto číslo je poté zaokrouhleno pomocí funkce $round(x)$ na nejbližší celé číslo.
Pokud má operace pouze jeden možný stroj pro přiřazení ($l_j = 1$), pak je vždy hodnota $r_j = 1$ a není potřeba provádět konverzi z hodnoty x_j.

$$r_j = round\left(\frac{1}{2\delta}(l_j - 1)(x_j + \delta) + 1\right), j = 1, 2, ..., d \tag{8}$$

Druhým krokem je převod druhé části vektoru $\vec{X}^{(2)} = [x_{d+1}, x_{d+2}, ..., x_{2d}]$, který je potřeba převést na vektor pro určení sekvence operací. Nejdříve se provede setřídění hodnot uvnitř vektoru $\vec{X}^{(2)}$ sestupně. Při tomto setřídění je potřeba si zapamatovat, na které pozici které číslo bylo, jelikož tato pozice hodnoty odkazuje na příslušnou operaci. Pozice s největší hodnotou ve vektoru $\vec{X}^{(2)}$ tedy určuje ID operace, která bude mít největší prioritu a bude do rozvrhu zařazena jako první. V úvahu je však potřeba také brát to, že každá sekvence operací není proveditelná, jak již bylo zmíněno v podkapitole 5.3.1. Je tedy ještě potřeba provést korekci pořadí operací v rámci dané zakázky podle výrobního postupu, pokud jsou tyto pravidla porušeny.

Pokud má vektor složení $\vec{X}^{(2)} = [0.8, -0.2, -0.75, 0.77, 0.98, -0.33, 0.12, -0.87, -0.15]$, je konkrétní konverze je znázorněna na obrázku 8. Operace stejné zakázky (Job) jsou označený stejnou barvou. Zakázka 1 je označena modrou barvou, zakázka 2 je označena barvou oranžovou a zakázka 3 je označena barvou zelenou. Červeně jsou pak označeny ID operací, které musí být v rámci své zakázky přeuspořádány, aby splňovaly pravidla výrobního postupu. Například v rámci práce 2 je potřeba ve vektoru $\vec{S}$ navzájem přehodit operace s indexy 4 (první operace v rámci



Obrázek 8: Konverze vektoru $\vec{X^{(2)}}$ na vektor $\vec{S}$

zakázky 2), 5 (druhá operace v rámci zakázky 2) a operace s indexy 6 (třetí operace v rámci zakázky 2), 7 (čtvrtá/poslední operace v rámci zakázky 2). Po těchto úpravách bude splněn správný pracovní postup této zakázky.

5.3.4 Zpětná konverze

Zpětná konverze znamená transformace dvou vektoru $\vec{R} = [r_1, r_2, ..., r_d]$ a $\vec{S} = [s_1, s_2, ..., s_d]$ do jednoho vektoru $\vec{X} = [x_1, x_2, ...x_d, x_{d+1}, ..., x_{2d}]$. Stejně jako dopřednou konverzi je potřeba také tuto zpětnou konverzi provádět ve dvou krocích.

V první části se provádí převod z vektoru přiřazení strojů $\vec{R}$ na první část vektoru $\vec{X}^{(1)} = [x_1, x_2, ..., x_d]$. K tomu slouží zpětná lineární transformace inverzní k lineární transformaci uvedené ve vzorci 8. Pokud má daná operace pouze jeden alternativní stroj, na kterém může být vykonávána, pak hodnota odpovídající této operaci ve vektoru $\vec{X}^{(1)}$ může být libovolná hodnota v rozsahu $[-\delta, \delta]$. Pokud je počet alternativních strojů $l_j \geq 1$, musí se hodnota r_j dané operace převést na hodnotu x_j pomocí vzorce 9.

$$x_j = \frac{2\delta}{l_j - 1}(r_j - 1) - \delta \tag{9}$$

V druhé části probíhá převod vektoru pro určení sekvence operací $\vec{S}$ tak, že se z vektoru $\vec{S}$ postupně od první až po poslední pozici odebírají hodnoty (ID operací). U tohoto procesu je potřeba mít k dispozici původní vektor $\vec{X}^{(2)}$ setříděný sestupně podle hodnot. Jakmile se

odebere první hodnota s_1 z vektoru $\vec{S}$, vezme se první nejvyšší hodnota ze setříděného vektoru $\vec{X}^{(2)}$. Tato hodnota se vloží do nového vektoru $\vec{X}^{(2)}$ na pozici, která odpovídá hodnotě s_1. Při odebrání druhé hodnoty s_2 z vektoru $\vec{S}$ se odebere druhá nejvyšší hodnota z původního setříděného vektoru $\vec{X}^{(2)}$. Tato hodnota je vložena na pozici odpovídající hodnotě s_2 v novém vektoru $\vec{X}^{(2)}$. Takto je potřeba projít celý vektor $\vec{S}$.

5.3.5 Evoluční proces

Samotný evoluční proces je sled po sobě jdoucích úkonů typických pro diferenciální evoluci. V první fázi algoritmu probíhá vytvoření počáteční populace. Vektor každého jedince je naplněn náhodnými reálnými čísly, které jsou korektní v rámci velikosti prohledávaného prostoru. Důležitým parametrem algoritmu je maximální počet populací, který určuje, kolikrát populace projde evolucí. V každé evoluci je vytvořena nová populace a je nalezen její nejlepší jedinec. 5 uvádí pseudokód tohoto evolučního procesu. Pokud se nejlepší jedinec 25 populací nezmění, je splněna ukončovací podmínka a evoluce je ukončena předčasně.

Algorithm 5 Pseudokód evolučního procesu

```
1:  function EvolutionProcess
2:      Vytvoření počáteční populace P
3:      Best ← Najít nejlepšího jedince v P
4:      počet populací ← 1
5:      while počet populací < maximální počet populací do
6:          P′ ← CreateNewPopulation()
7:          Best ← Najít nejlepšího jedince v P′
8:          Zjistit zlepšení populace
9:          počet populací ← počet populací + 1
10:         if Pokud je splněna ukončovací podmínka then return Best
11:         end if
12:     end while
          return Best
13: end function
```

5.3.6 Vytvoření nové populace

V každé evoluci se musí vytvořit nová populace, která nahradí stávající populaci do další evoluce. Tato nová populace je následně naplněna jedinci, kteří vycházejí z populace původní. Tito jedinci jsou modifikováni pomocí fází evoluce - mutací a křížením. Pro každého jedince I jsou náhodně vybráni dva jedinci z populace a je vytvořen tkz. zkušební vektor T. Nyní pro každý parametr (gen) je vytvořen parametr v jedinci T pomocí mutace zadané vzorcem 5, ve kterém hraje roli vstupní mutační parametr algoritmu F. Po mutaci vždy probíhá kontrola, zda je parametr korektní a spadá do prohledávaného prostoru. V případě, kdy parametr korektní není, probíhá korekce dle pseudokódu 6. Vzorce opravující tyto parametry byly převzaty z [14].

Jakmile je parametr korektní v rámci prohledávaného prostoru, přichází na řadu proces křížení. Křížení probíhá na základě náhodně vygenerovaného čísla a vstupního parametru pravděpodobnosti křížení Cr podle vzorce 6. Jakmile jsou vytvořeny všechny parametry uvnitř zkušebního jedince T, je potřeba vypočítat jeho hodnotu fitness. Na základě této hodnoty je v dalším kroku prováděna selekce mezi tímto jedincem T a původním jedincem I. Lepší z těchto dvou jedinců je zařazen do další/nové populace. Slovem lepší je myšleno vyhodnocení na základě hodnoty fitness a význam porovnání těchto hodnot se liší na základě toho, jestli je řešena úloha na maximalizaci či minimalizaci. Tento postup je vyjádřen pseudokódem 7.

Algorithm 6 Pseudokód korekce parametru jedince

```
1: if hodnota parametru < spodní hranice then
2:     Hodnota parametru ← spodní hranice + (horní hranice - spodní hranice) * rand(0,1)
3: end if
4: if hodnota parametru > horní hranice then
5:     Hodnota parametru ← horní hranice - (horní hranice - spodní hranice) * rand(0,1)
6: end if
```

Algorithm 7 Pseudokód vytvoření nové populace

```
1: function CreateNextPopulation
2:     Vytvoření nového populace P'
3:     for all jedince I v jedincích populace P  do
4:         A,B ← Vyber z populace P náhodně dva jedince
5:         Vytvoř nového jedince T
6:         for i ← 1, počet parametrů v jedinci I do
7:             T[i] ← Pro parametr i v jedinci T proveď mutaci
8:             Kontrola a korekce parametru T[i]
9:             Křížení parametru T[i] s parametrem I[i]
10:        end for
11:        Zjištění hodnoty fitness u jedince T
12:        Selekce mezi jedinci T a I pro další populaci P'
13:    end forreturn P'
14: end function
```

5.4 Reprezentace výrobního rozvrhu

Reprezentací výrobního rozvrhu se rozumí určitá forma zápisu posloupnosti operací na daných strojích do určité struktury. Tato struktura umožní provádět výpočty nad daným rozvrhem. V této práci je nutné vypočítat fitness funkce či jiné optimalizační výpočty sloužící k optimalizaci výrobního rozvrhu. S postupem několika posledních let vzniklo několik forem reprezentace výrobních rozvrhů. Nejpoužívanější reprezentace jsou pomocí disjunktivního grafu nebo preferenčního seznamu. Pro tuto práci však bude výhodnější použití reprezentace disjunktivním grafem, protože tato metoda se často jeví jako výhodnější při používání heuristických metod.

5.4.1 Disjunktivní graf

Disjunktivní graf G je definován množinou uzlů V, množinou konjunktivních hran C a množinou disjunktivních hran D, $G = (V, C \cup D)$. Množina úloh V reprezentuje všechny operace všech pracovních úloh, které je potřeba rozvrhovat. Navíc tato množina obsahuje dva speciální uzly. První z těchto uzlů je uzel S, který se nazývá startovní uzel. Druhým speciálním uzlem je uzel E nazývající se koncový uzel. Všechny uzly kromě těchto dvou speciálních uzlů mají určité ohodnocení. Toto ohodnocení značí trvání odpovídající operace v minutách. Z hlediska konjunktivních hran C a disjunktivních hran D se jedná v obou případech o orientované hrany. Orientované konjunktivní hrany spojují dva uzly (operace), které jsou prováděny v rámci jedné pracovní úlohy a vyjadřují jejich pořadí. Důležité jsou také hrany, které nám spojují startovní uzel S se všemi prvními uzly (operacemi) všech pracovních úloh. Taktéž z posledního uzlu všech pracovních úloh musí vést konjunktivní hrana do koncového uzlu E. Prvními uzly jsou myšleny operace, které musí být v rámci své pracovní úlohy vykonávány jako první. Posledními uzly jsou myšleny operace, které musí být v rámci své pracovní úlohy vykonávány jako poslední.

Disjunktivní hrany spojují operace vykonávající se na stejném stroji. Množiny disjunktivních hran D vytváří množinu M úplných podgrafů, kde počet podgrafů je ekvivalentní počtu strojů. Každý podgraf je rozvrhem pro určitý stroj.

5.4.2 Topologické uspořádání grafu

Předtím, než je možné v grafu hledat kritickou (nejdelší) cestu, je potřeba, aby graf byl topologicky uspořádaný. Topologické uspořádání uzlů orientovaného acyklického grafu je taková posloupnost uzlů $v_1, v_2, ..., v_n$, kde každá hrana mezi uzly vede zleva doprava. Jinak řečeno pokud máme orientovanou hranu z uzlu v_i do uzlu v_j, tak musí platit, že $i < j$. Z této definice vyplývá, že prvním uzlem musí být uzel, do kterého nevede žádná hrana. V řešení této práce se jedná o startovní uzel S. Algoritmus pro topologické uspořádání grafu je podobný algoritmu prohledávání do hloubky a liší se v několika detailech. Algoritmus prohledávání do hloubky postupuje tím, že vytiskne vrchol a až poté se rekurzivně volá opět tento algoritmus pro jeho přilehlé vrcholy. Při algoritmu topologického uspořádání je důležitou součástí datová struktura zásobník. Místo toho, aby se tento vrchol ihned vytiskl/označil, nejprve je zavolán rekurzivně stejný algoritmus topologického třídění na jeho přilehlé vrcholy. Až poté je vrchol umístěn do zásobníku. Každý vrchol by měl být do zásobníku umístěn až v tu chvíli, kdy všechny jeho přilehlé vrcholy (i jejich přilehlé vrcholy atd.) již v zásobníku jsou. Když se následně budou vrcholy ze zásobníku odebírat, orientované hrany mezi těmito vrcholy budou směřovat pouze zleva doprava. Pokud by během rekurzivního volání algoritmu pro topologické uspořádání byl navštíven některý vrchol více než jednou, znamenalo by to, že v grafu byl nalezen cyklus. V tomto případě je algoritmus ošetřen výjimkou - ukončením této funkce. Pokud je tento algoritmus ukončen úspěšně, pak je zaručeno, že daný graf má topologické uspořádání, neboli že je acyklický.

Algorithm 8 Pseudokód topologického uspořádání grafu

```
1: function TopologicalOrdering(node)
2:     for all následovníky n v node do
3:         if n ještě nebyl navštíven then
4:             TopologicalOrdering(n)
5:         end if
6:     end for
7:     if node ještě nebyl navštíven then
8:         Dát node do zásobníku
9:         Označit node jako navštívený
10:    end if
11: end function
```

5.4.3 Kritická cesta v grafu

Algoritmus kritické cesty je algoritmus sloužící k plánování množiny aktivit určitého procesu. V této práci se konkrétně jedná o rozvrhování operací pro výrobní proces. Kritickou cestou se rozumí nejdelší možná cesta ze startovního uzlu S do koncového uzlu E disjunktivního grafu. Každý disjunktivní graf má minimálně jednu kritickou cestu. Těchto kritických cest však může být i více. Každá kritická cesta se skládá z několika operací, které určují časový rámec daného rozvrhu. Čas ukončení poslední operace, která se vyskytuje na kritické cestě, určuje výsledný čas celého rozvrhu. Všechny operace, které se nacházejí na kritické cestě mají nulovou časovou rezervu. Nulovou časovou rezervou se rozumí, že pokud se tyto operace opozdí a nezačnou podle plánu, má to vliv na výsledný čas dokončení celého rozvrhu - dojde k pozdějšímu dokončení.

Pokud tedy existuje vytvořený acyklický orientovaný graf, může na něm být spuštěn algoritmus k nalezení kritické cesty. Každý uzel v grafu má časové ohodnocení v minutách, kromě uzlů S a E. Tyto dva uzly mají nulové ohodnocení. U každého uzlu v grafu je nutné pro tento algoritmus mít dvě proměnné s názvy **levá** a **pravá**. Obě tyto proměnné jsou datového typu číslo. Tento algoritmus se skládá ze dvou funkcí. První funkce zajišťuje průchod grafem ze startovního uzlu S do koncového uzlu E zleva doprava. V tomto průchodu se postupně vyplňuje u každého uzlu proměnná *levá*. Druhá funkce prochází graf opačným směrem zprava doleva a vyplňuje se proměnná *pravá*.

Princip první funkce zleva doprava spočívá v tom, že do proměnné *levá* se zapisuje hodnota proměnné *levá* předchozího vrcholu plus ohodnocení aktuálního vrcholu. Průchod mezi vrcholy může být uskutečněn přes konjunktivní i disjunktivní hranu. Při vstupu do vrcholu je vybrána ta příchozí hrana, ze které bude nejvyšší hodnota proměnné *levá*. Takto se vyplní postupně všechny vrcholy až do koncového vrcholu E. Jakmile se algoritmus dostane do poslední vrcholu E, máme v jeho proměnné *levá* minimální délku celého rozvrhu.

Princip druhé funkce procházející graf zprava doleva pracuje s proměnnou *pravá*. Na začátku se v koncovém vrcholu E opíše do proměnné *pravá* hodnota z proměnné *levá*. Nyní se postupuje proti směru orientovaných hran. Do následujících vrcholů se zapisují hodnoty do proměnných

pravá. Tato hodnota je vždy získána z hodnoty proměnné *pravá* předchozího vrcholu (směrem zprava doleva) mínus ohodnocení tohoto předchozího vrcholu. Nyní se však při výběru hrany vybírá taková hrana, aby byla získána co nejmenší hodnota proměnné *pravá*. Takto se projde celý graf až do počátečního vrcholu S. V tomto vrcholu by měla být hodnota proměnné *pravá* 0. Ty vrcholy, které mají hodnotu proměnné *levá* i *pravá* rovny, se nacházejí na kritické cestě. Znovu je potřeba zdůraznit, že graf může mít jednu nebo více kritických cest.

Algorithm 9 Pseudokód nalezení kritické cesty v grafu

```
1: for all vrcholy n v grafu do
2:     Označit n jako nenavštívený vrchol
3: end for
4: Topologicky uspořádat graf
5: Projít graf zleva doprava
6: Projít graf zprava doleva
7: for all vrcholy n v grafu do
8:     Zjistit zda vrchol n leží na kritické cestě
9: end for
```

5.4.4 Příklad nalezení kritické cesty

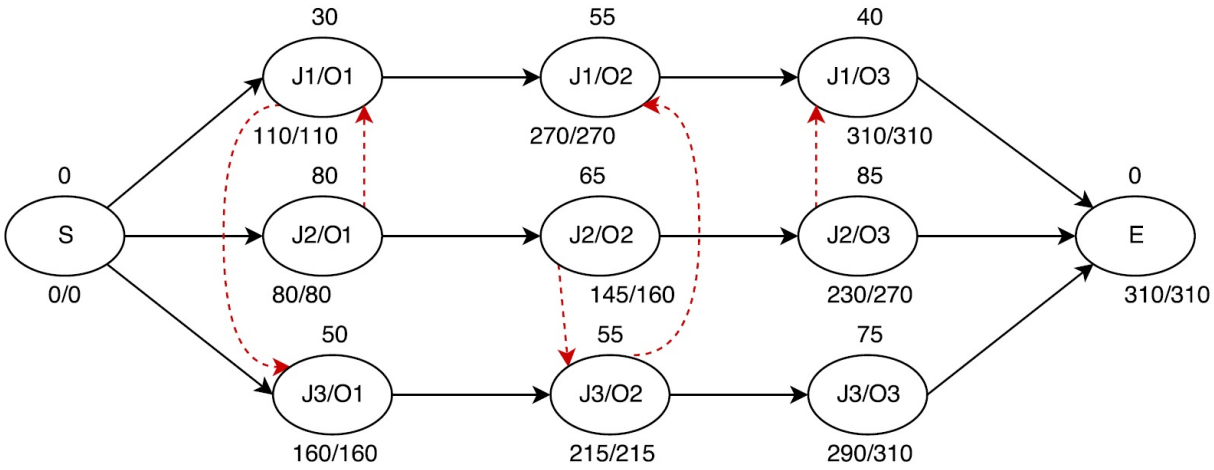
Na obrázku 9 je znázorněn příklad, kde se rozvrhují 3 zakázky. Každá ze zakázek se skládá ze tří operací. Je daný disjunktivní graf, který znázorňuje výrobní rozvrh pro tyto zakázky. Konjunktivní hrany jsou znázorněny plnou čarou a znázorňují pořadí operací na dané zakázce. Disjunktivní hrany jsou vyjádřeny přerušovanou čarou a udávají pořadí operací na stroji. Z příkladu je zřetelné, že operace J2/O1 → J1/O1 → J3/O1 budou prováděny na stejném stroji a to v uvedeném pořadí.

V první fázi musí být provedeno topologické uspořádání uzlů v grafu. Topologické uspořádání tohoto grafu je: S, J2/O1, J2/O2, J2/O3, J1/O1, J3/O1, J3/O2, J3/O3, J1/O2, J1/O3, E a je zachyceno obrázkem 10. Zadaný graf je na obrázku 9. Na tomto příkladu lze vidět, že po topologickém uspořádání hrany směřují pouze zleva doprava.

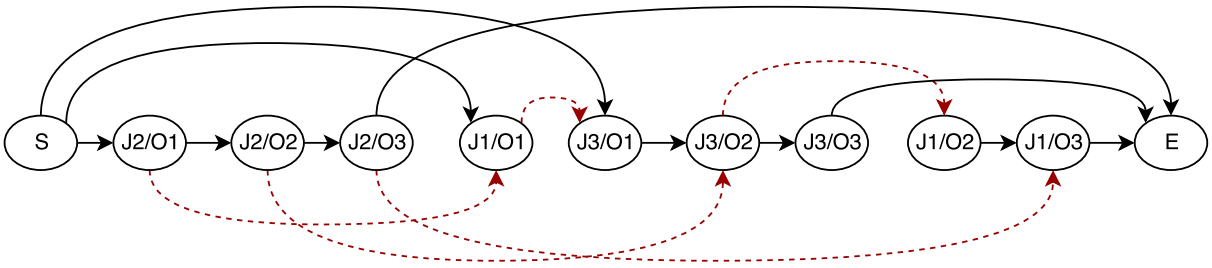
Nyní může být zahájen algoritmus kritické cesty, přičemž vrcholy budou procházeny zleva doprava i v opačném směru podle topologického uspořádání. Operace J1/O1 má hodnotu proměnné *levá* 110. Do této operace se však lze dostat z vrcholu S i z vrcholu $J2/O1$. Podle pravidla algoritmu se však vybírá nejvyšší možná hodnota proměnné *levá*. Pokud by se přišlo po hraně z vrcholu S, byla by hodnota proměnné *levá* 30. Pokud se přijde z vrcholu $J2/O1$ je hodnota rovna 110 (80 + 30). To znamená, že je vybrána varianta, ve které se do vrcholu $J1/O1$ přijde z vrcholu $J2/O1$.

5.5 Účelové funkce

Posledním důležitým parametrem pro problém rozvrhování je optimalizační kritérium. Optimalizace daného rozvrhu může být posuzována podle mnoha různých faktorů. Níže jsou vypsány



Obrázek 9: Příklad disjunktivního grafu



Obrázek 10: Topologické uspořádání grafu

implementované optimalizační kritéria pro problém rozvrhování.

5.5.1 Maximální čas dokončení zakázek

Cílem této optimalizace je co nejvíce minimalizovat hodnotu C_{max}, což je celkový čas rozvrhu. Pseudokód 10 vystihuje tuto účelovou funkci.

Algorithm 10 Pseudokód optimalizace maximální čas dokončení zakázek

```
1: function OptimalizationCmax
2:     C_max ← Hodnota left posledního uzlu v zásobníku
3: return C_max
4: end function
```

5.5.2 Priorita zakázek

Při této optimalizaci by měly být zohledněny priority jednotlivých zakázek. U těchto zakázek je nutné, aby měly zadanou určitou váhu w a výrobní čas C. Cílem této optimalizace je, aby celkový výrobní čas C_{max} byl co nejvíce minimalizován. Zároveň je žádoucí, aby zakázky s vyšší prioritou byly v rozvrhu vykonány dříve, než zakázky s prioritou nižší. Při této optimalizaci je využit vzorec $\Sigma\ w_j\ C_j$ a postup je vyjádřen níže 11.

Algorithm 11 Pseudokód optimalizace priority zakázek

```
1: function OptimalizationByPriority
2:     for all uzly n v předchozích uzlech koncového uzlu E do
3:         Výrobní čas t ← t * priorita/váha zakázky w
4:     end for
5:     Přepočítej levé a pravé hodnoty všech uzlů v grafu
6:     C_max ← Hodnota left posledního uzlu v zásobníku
7: return C_max
8: end function
```

5.5.3 Maximální doba zpoždění

Optimalizace maximální doby zpoždění má za cíl minimalizovat opoždění zakázek. K této optimalizaci je potřeba znát dodací termín zakázky d_i. Postup je vyjádřen pseudokódem 12.

5.5.4 Celková proudová doba výroby

Touto optimalizací se rozumí taková doba průtoku zakázky J ve výrobě, po kterou je zakázka J v procesu výroby. Je to doba mezi tím, kdy byla zakázka vpuštěna do výroby a doba dokončení zakázky. Cílem této minimalizace je, aby byla zakázka co nejdříve hotová, pokud je vpuštěna do výroby. Postup je vyjádřen níže.13

Algorithm 12 Pseudokód optimalizace maximálního zpoždění

```
1: function OptimalizationByTardiness
2:     tardiness ← 0
3:     for all uzly n v předchozích uzlech koncového uzlu E do
4:         tardiness ← tardiness + max(hodnota left v uzlu n - nejpozdější termín dokončení
   zakázky , 0)
5:     end for
       return tardiness
6: end function
```

Algorithm 13 Pseudokód optimalizace proudové doby výroby

```
1: function OptimalizationFlowTime
2:     for all uzly n v následujících uzlech počátečního uzlu S do
3:         Každému uzlu n nastav datum uvolnění zakázky r_n
4:     end for
5:     flowTime ← 0
6:     for all uzly n v předchozích uzlech koncového uzlu E do
7:         flowTime ← flowTime + Hodnota left - datum uvolnění zakázky r_n
8:     end for
       return flowTime
9: end function
```

5.6 Návrh modulů

Tato část je věnována návrhu dvou modulů. Každý modul řeší odlišnou část implementace.

5.6.1 Návrh modulu řešící rozvrhování

Název modulu, který řeší samotný algoritmus a logiku rozvrhování, je „ImproveIT.CapacityPlanning“. Tento modul se skládá z potřebných struktur, které mohou být rozděleny do několika následujících kategorií.

- Struktury řešící job shop problém
- Struktury k práci s disjunktivním grafem
- Struktury udržující datovou sadu
- Struktury k využití evolučního algoritmu
- Struktury pro převod dat ze struktur normy ISA 95 do struktur job shop problému

Všechny tyto struktury jsou implementovány a mají mezi sebou různé asociační vztahy. Třídní diagram zachycující všechny tyto struktury je k nahlédnutí v neveřejné části této práce.

5.6.2 Modul pro funkcionalitu uživatelského rozhraní

Pro implementaci funkcionality uživatelského rozhraní byl vytvořen zvláštní modul „ImproveIT.CapacityPlan
Tento modul zajišťuje, aby uživateli byl v aplikaci iMPROVE iT! zobrazen kompletní výrobní plán. Pokud uživatel chce, aby byl výrobní rozvrh znovu přepočítám evolučním algoritmem, tento modul aktivuje algoritmus rozvrhování, který se nachází v modulu „ImproveIT.CapacityPlanning“. Všechny funkce a možnosti, které tento modul poskytuje budou více představeny v kapitole 6.4.2.

6 Integrace modulů do systému iMPROVE iT!

Tato kapitola se věnuje samotné integraci modulů kapacitního plánování do systému iMPROVE iT!, který je zde taktéž představen. Integrace se týká dvou modulů vytvořených v této práci, kterými jsou „ImproveIT.CapacityPlanning“ a „ImproveIT.CapacityPlanning.Win“.
Modul „ImproveIT.CapacityPlanning“ zahrnuje potřebné struktury a veškerý algoritmus kapacitního plánování. Druhý modul „ImproveIT.CapacityPlanning.Win“ slouží k interpretaci naplánovaných dat do tlustého klienta běžícího pod systémem Windows. Tento modul zahrnuje vizualizační komponenty a metody pro ovládání uživatelského rozhraní.

6.1 Systém iMPROVE iT!

iMPROVE iT! společnosti SCADA Servis je modulární MES/MOM systém, který umožňuje realizovat funkce výrobního informačního systému. Tento software zpracovává data a poskytuje informace uživatelům v reálném čase. To jim umožňuje okamžitou reakci na události, které ve výrobě vznikají. Implementovaný modul kapacitního plánování, kterému je věnována tahle práce, je dalším modulem a zároveň rozšířením tohoto systému.

Tento systém je postavený na eXpressApp frameworku (XAF), což je univerzální aplikační framework umožňující vytvářet byznys aplikace běžící jak na webu tak i na Windows. Tento framework usnadňuje implementaci bohatých funkcí a vysoce interaktivních aplikací založených na „DevExpress Windows Forms“ a „ASP.NET“ formulářů a ovládacích prvků. Jako ORM mezi aplikační pamětí a databází lze využít Entity Framework nebo DevExpress XPO knihovny. Mezi největší výhody tohoto frameworku patří automatické vytváření uživatelského rozhraní a rozhraní databáze.

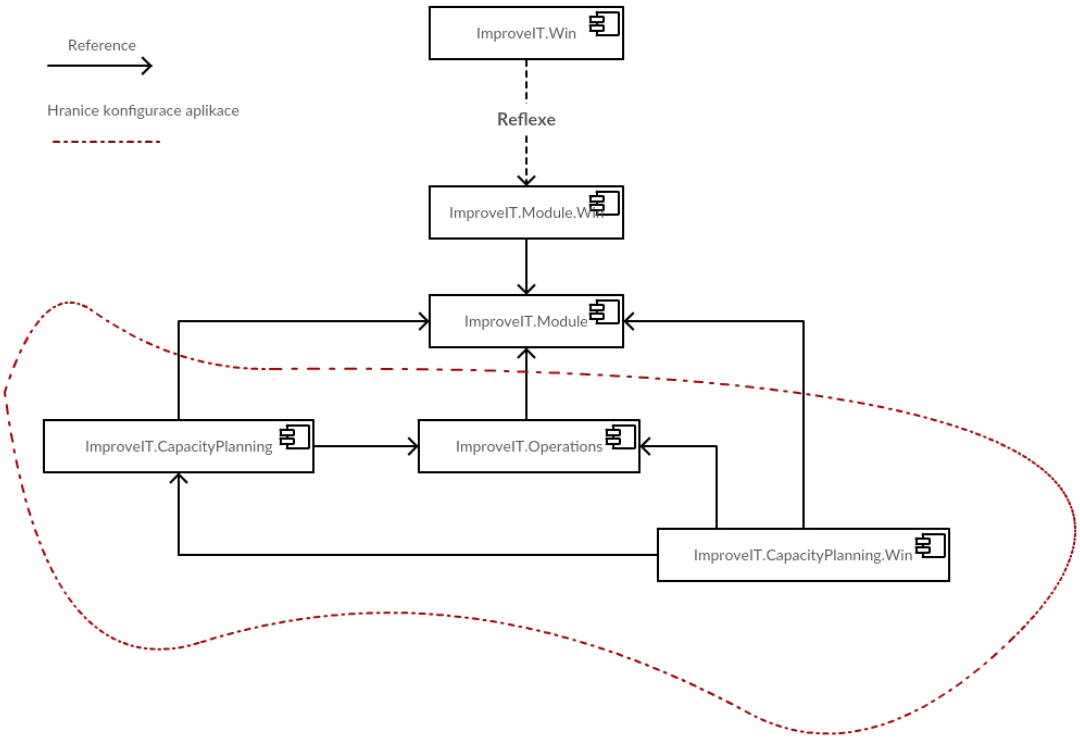
Aby byl modul pro kapacitní plánování použitelný v systému iMPROVE iT!, musí být vytvořen jako modul postavený na eXpressApp frameworku.

6.2 Vztah modulů k iMPROVE iT!

Jak již bylo řečeno systém iMPROVE iT! je modulární systém, který je postaven na třech základních modulech. Tyto tři základní moduly tvoří čistou kostru systému a samy o sobě neobsahují žádnou velkou funkčnost. Základní modul „ImproveIT.Module“ obsahuje všechny obecné byznys objekty pro celou aplikaci. Modul pro kapacitní plánování musí mít referenci na tento základní modul. Jelikož tento modul pro plánování potřebuje samotné data z byznys procesu výroby, které se nacházejí v modulu „ImproveIT.Operations“, je potřeba na tento modul taktéž mít referenci. Struktury a následné ukládání dat těchto byznys objektů v modulu „ImproveIT.Operations“ jsou postaveny na standardu ANSI/ISA 95, který je více zmíněn v další části práce.

Architektura systému je naznačena diagramem komponent 11. Tento diagram zachycuje všechny reference mezi moduly. Důležitou roli hraje hranice konfigurace aplikace, která vyznačuje moduly nacházející se v nastavení konfiguraci aplikace. Tato konfigurace je po startu

aplikace načtena a všechny moduly nacházející se této konfiguraci jsou načteny pomocí reflexe. Moduly „ImproveIT.Module“ a „ImproveIT.Module.Win“ jsou pomocí reflexe načteny vždy automaticky.



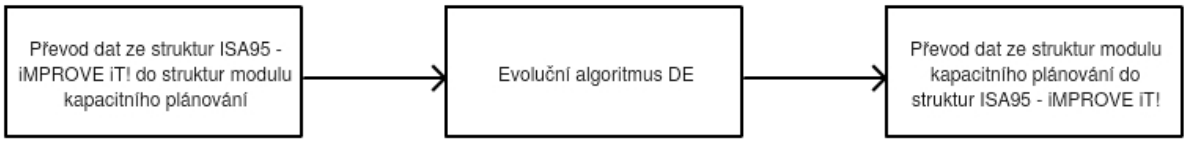
Obrázek 11: Diagram komponent

6.3 ANSI/ISA 95

ANSI/ISA 95 (dále jen ISA 95) je mezinárodní standard pro integraci podnikových a výrobních systémů. Tento standard byl vytvořen globální neziskovou organizací ISA. ISA-95 se skládá z modelů a terminologie. Ty mohou být používány k určení toho, které informace musí být vyměňovány mezi systémy pro prodej, finance, logistiku a systémy pro výrobu, údržbu či kvalitu. Tato informace je strukturována do UML modelů, které jsou základními prvky pro vývoj standardních rozhraní mezi ERP a MES systémy. Tento standard může být použit pro různé účely, například jako vodítko pro definici uživatelských požadavků, pro výběr MES dodavatele a jako základ pro vývoj MES systémů a databází.[5]

ISA-95 je mezinárodní standard pro vývoj automatizovaného rozhraní mezi podnikovými a řídícími systémy. Tato norma byla vyvinuta pro světové výrobce. Slouží pro aplikace ve všech odvětvích a všech typech procesů, jako jsou dávkové procesy, kontinuální a opakující se procesy.[5]

Cílem ISA-95 je poskytnout konzistentní terminologii, která je základem pro dodavatele a výrobní komunikaci poskytující konzistentní informační modely a konzistentní modely činností. Ty jsou základem pro objasnění funkčnosti aplikace a toho, jak má být určitá informace využita.[5]



Obrázek 12: Rozvrhování v iMPROVE iT!

Hlavním cílem, proč je modul „ImproveIT.Operations“ ve standardu ISA 95, je zajištění kvalitní a bezproblémové komunikace se všemi ERP systémy podporujícími toto rozhraní. Aby bylo možné data z modulu „ImproveIT.Operations“ převést z formy ANSI/ISA 95 do modelů modulu kapacitního plánování či zpět, pak je potřeba detailního nastudování a porozumění modelu ANSI/ISA 95. Postup, jak probíhá samotný proces rozvrhování v kontextu převodu dat z i do ISA 95 struktur, je znázorněno obrázkem 12.

ISA-95 standard se skládá z 5 částí uvedených níže. Některé tyto standardy budou představeny avšak pouze ve zkratce, jelikož tento standard je sám o sobě velice obsáhlý.

1. ANSI/ISA-95.00.01 Modely a terminologie
2. ANSI/ISA-95.00.02 Objekty a atributy
3. ANSI/ISA-95.00.03 Modely aktivit
4. ANSI/ISA-95.00.04 Objekty a atributy pro MOM
5. ANSI/ISA-95.00.05 Transakce mezi výrobními a ekonomickými systémy

6.3.1 ANSI/ISA-95.00.01 Modely a terminologie

První část standardu definuje rozhraní mezi podnikovými a výrobními systémy. Jsou to rozhraní mezi úrovněmi 3 a 4 hierarchického modelu 13 definovaného tímto standardem. Hlavním cílem je zvýšit jednotnost a konzistentnost rozhraní, snížit rizika, cenu a chyby spojené s implementací těchto rozhraní. Tato norma slouží ke snížení úsilí spojených s implementací nových produktů. Hlavním cílem je, aby podnikové a kontrolní systémy, které fungují současně a navzájem se ovlivňují, bylo snadné integrovat. [6]
Tato část standardu obsahuje:

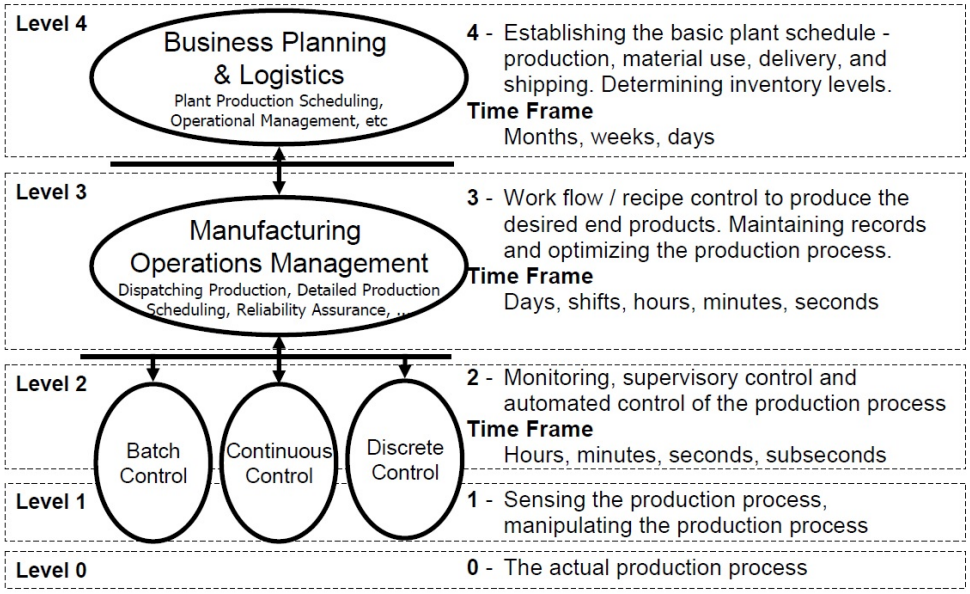
(a) Definice rozsahu výrobních činností

(b) Fyzické aktiva podniku zapojených do výroby

(c) Výpis funkcí spojených s rozhraním mezi řídícími funkcemi a funkcemi podniku

(d) Popis informací, které jsou sdílené mezi řídícími a podnikovými funkcemi

Obrázek 13 znázorňuje různé úrovně funkčního modelu hierarchie.



Obrázek 13: Funkční hierarchie ISA 95, převzato z [6]

- Úroveň 0 definuje skutečné fyzikální procesy.
- Úroveň 1 jsou činnosti spojené se snímáním fyzikálních procesů. Nejčastěji se jedná o čidla, senzory atd.
- Úroveň 2 definuje činnosti monitorování a kontrolu fyzikálních procesů, které pracují v jednotkách hodin, minut, sekund a milisekund.
- Úroveň 3 typicky působí v rámci dnů, směn, hodin, minut a sekund. V této úrovni se definují toky činností k vytvoření požadovaného produktu.
- Úroveň 4 definuje činnosti potřebné k řízení výroby organizace (stanovení zásob, plány, dodávky materiálů). V této úrovni je nejčastější časové měřítko v rámci měsíců, týdnů a dnů.

Důležitým faktem je, že hranice mezi úrovní 3 a 4 není pevně definovaná a je velice pohyblivá dle konkrétních případů. V některých případech aktivity úrovně 3 můžou zasahovat do úrovně 4 a může tomu být i naopak.

V této práci na úrovni 4 stojí ERP systém, který systému iMPROVE iT! pracujícímu na úrovni 3 poskytuje dané zakázky, definice kapacit a výrobní postupy. Systém ERP tedy má hrubý plán zakázek naplánovaných v rámci měsíců a týdnů. Pro tyto zakázky bude následně systémem iMPROVE iT! vytvořen detailní optimální výrobní rozvrh. K tomu slouží modul vytvořený v této práci.

6.3.2 ANSI/ISA-95.00.02 Objekty a atributy

Tato část standardu se věnuje na definování objektových modelů a atributů vyměňovaných informací, které jsou definované v části 1. Pro tuto práci je potřeba znát model pro stroje, definice výrobních postupů a definice zakázek. Všechny tyto tři potřebné modely jsou k nahlédnutí v přílohách A.1, A.2 a A.3.

Tato část standardu obsahuje spoustu dalších modelů, které však není potřeba v této práci detailně rozebírat.

6.4 Zobrazení výrobního rozvrhu

Zobrazení výrobního rozvrhu uživateli je realizováno pomocí Ganttova diagramu.

6.4.1 Ganttův diagram

Ganttův diagram je grafické znázornění rozvržených či naplánovaných aktivit v čase. Využívá se při řízení projektů, výroby a při dalších různých činnostech. Tento diagram se skládá ze sloupců, které značí časové období. Tyto sloupce mohou být různého časového horizontu (minuty, hodiny, týdny, měsíce) v závislosti na délce rozvrhovaných aktivit. V řádcích se zobrazují dílčí aktivity nebo úkoly v odpovídajícím pořadí, v jakém odpovídá jejich sled vykonávání.

V této práci je Ganttův graf použitý na zobrazení rozvržených operací v čase. Každý jednotlivý řádek značí stroj ve výrobě. Časovou horizontální osu lze nastavit v rámci minut - dnů. Jednotlivé operace jsou umístěny na řádku značící přidělený stroj, kde se tato operace má vykonávat. Jako komponenta Ganttova grafu je využita komponenta třetí strany již zmiňovaného XAF frameworku.

6.4.2 Možnosti uživatelského rozhraní

Uživatelské rozhraní pro rozvrhování je rozděleno na dvě poloviny. V horní části obrazovky se nachází samotný Ganttův diagram, kde se nachází rozvržené zakázky. V dolní polovině lze vidět zakázky, které jsou uvolněné k rozvrhování, ale zatím rozvržené nejsou.

Toto rozhraní umožňuje doposud nerozvrženou zakázku přetáhnout z tabulky zakázek do Ganttova diagramu díky implementované funkci „Drag and Drop“ v modulu „CapacityPlanning.Win“. V tento moment je uživatel dotázán, jestli chce tuto novou zakázku umístit do Ganttova diagramu úplně nakonec, za již rozvržené zakázky, nebo zda chce celý rozvrh znovu rozvrhnout pomocí algoritmu diferenciální evoluce. Proces celého přeplánování zahrnuje ty operace, které ještě nezačaly a nezačínají dříve než je proces přeplánování spuštěn plus jednu hodinu.

Významnou funkcí tohoto uživatelského rozhraní je také to, že pokud uživatel klikne na libovolnou operaci, tato operace se označí určitou barvou. Navíc se stejnou barvou označí všechny ostatní operace patřící stejné zakázce jako tato označená operace. Navíc dojde k propojení těchto

operací šipkami, které označují trasu a posloupnost operací v rámci zakázky.

Algorithm 14 Pseudokód přetažení nové zakázky na konec existujícího rozvrhu

1: Načíst všechny operace z dané zakázky
2: **for all** operace o ve všech operacích zakázky **do**
3: Zjistit všechny možné stroje, na kterých může být operace o vykonána
4: Vybrat stroj, kde bude operace o vykonána nejdříve
5: Umístit operaci o do rozvrhu tím způsobem, aby splňovala podmínku, že čas začátku operace o je větší než čas ukončení předchozí operace
6: **end for**

7 Testování modulu na reálných datech

Tato kapitola se věnuje testování algoritmu na konkrétních reálných datech. Tabulka 8 představuje velikost dat, nad kterými bude spuštěn algoritmus rozvrhování. Testování bude taktéž probíhat pro různé hodnoty vstupních řídících parametrů algoritmu diferenciální evoluce. Výsledky pro různé parametry budou mezi sebou porovnány a bude určeno nejlepší nastavení tohoto algoritmu pro zadaný problém.

7.1 Testování dle řídících parametrů algoritmu

Algoritmus diferenciální evoluce má hned několik parametrů, které ovlivňují rychlost a kvalitu evoluce. Toto testování bylo provedeno nad menší datovou sadou. Cílem tohoto testu není naplánovat všechny zakázky, ale zjistit jak se algoritmus chová při různých řídících parametrech. Velikost datové sady pro toto testování lze vidět v tabulce 4.

Ze získaných výsledků lze usoudit, že parametry $F = 0.2, Cr = 0.6$ a $F = 0.2, Cr = 0.8$ patří mezi mezi nelepší nastavení řídících parametrů algoritmu. Jako výchozí stav bude u algoritmu nastavená varianta $F = 0.2, Cr = 0.8$. Toto nastavení řídících parametrů také potvrdilo doporučené hodnoty, které lze vyčíst v mnoha odborných literaturách zabývajících se touto problematikou. Velikost populace v těchto testech nehrála příliš vysokou roli, jelikož byly výsledky vždy velice podobné. Je však nutné zdůraznit, že algoritmus využívá prvky náhody a výsledné řešení je pokaždé jiné. Hodnoty v tabulkách znamenají hodnoty C_{max} v minutách. Prázdné buňky v tabulce znamenají, že algoritmus do této populace vůbec nedošel, jelikož evoluce „uvízla“ v lokálním minimu. Z tohoto minima se už algoritmus nedostal a byl předčasně ukončen.

Počet zakázek	29
Počet operací	144

Tabulka 4: Velikost datové sady pro testování řídících parametrů

Testování bylo provedeno pro různé kombinace parametrů Cr a F. Doba celého algoritmu byla omezena na maximálně počet 400 populací. Tyto kombinace byly testovány při velikosti 500 jedinců - tabulka 5, 250 jedinců - tabulka 6 a 1 000 jedinců - tabulka 7. Každé toto testování bylo provedeno opakovaně několikrát a tabulka zachycuje průměrné hodnoty všech těchto měření.

Populace	F=0.2, Cr=0.2	F=0.2, Cr=0.6	F=0.2, Cr=0.8	F=0.4, Cr=0.2	F=0.4, Cr=0.8	F=0.6, Cr=0.2	F=0.6, Cr=0.8	F=0.8, Cr=0.2	F=0.8, Cr=0.8
1.	24 864	22 780	23 456	24 174	23 548	24 054	23 664	22 885	25 112
5.	22 117	22 142	22 086	21 545	23 249	23 423	23 664	22 298	22 586
10.	22 117	19 153	21 221	21 518	22 786	21 309	22 378	22 170	21 780
20.	21 533	18 541	19 974	21 518	21 845	21 309	20 759	21 757	21 780
30.	20 800	18 068	19 183	20 853	21 845	21 309	20 759	21 328	20 159
50.	20 800	17 714	18 804	20 853	21 845	21 309	20 626	21 318	20 159
80.	20 800	17 714	18 804	20 312	21 520	21 309	20 326	21 318	20 043
100.	20 800	17 714	18 804	19 844	21 520	20 089	20 326	21 318	19 881
130.	20 800	17 714	18 804	19 844	21 207	20 089	20 326	20 879	19 653
160.				19 844	20 442	20 089	20 326	20 879	19 612
190.				19 844	20 442	20 089		20 879	19 612
220.					20 292			20 879	19 609
250.					20 292				19 518
300.					20 292				18 753
350.					20 292				18 468
400.									18 468

Tabulka 5: Testování řídících parametrů pro velikost populace 500 jedinců

Populace	F=0.2, Cr=0.2	F=0.2, Cr=0.6	F=0.2, Cr=0.8	F=0.4, Cr=0.2	F=0.4, Cr=0.8	F=0.6, Cr=0.2	F=0.6, Cr=0.8	F=0.8, Cr=0.2	F=0.8, Cr=0.8
1.	24 654	23 855	25 131	24 594	26 251	24 793	25 313	25 534	24 460
5.	22 193	23 108	22 676	23 410	23 170	23 513	22 820	23 632	22 928
10.	22 193	23 108	21 628	23 138	22 222	22 095	22 820	22 994	22 494
20.	20 673	22 759	19 867	22 102	21 998	22 095	20 640	22 196	22 125
30.	20 673	22 226	19 688	21 553	20 357	21 197	20 640	21 306	22 125
50.	20 673	21 233	19 551	21 553	20 357	21 182	20 640	21 306	20 340
80.	20 673	21 233	19 551	20 998	20 357	21 182	20 620	21 048	20 263
100.	20 673	20 672	19 551	20 998	19 894	20 506	19 964	21 318	20 263
130.		20 672	19 551	20 998	19 665	20 506	19 964	21 318	20 263
160.		20 672		20 998	19 593	20 159	19 964	20 612	20 263
190.		20 571			19 593	20 159	19 964	20 492	
220.		20 014			19 152	19 932		20 135	
250.		19 405			19 152	19 932		19 417	
300.		18 841			19 152	19 932		19 373	
350.		18 637						19 373	
400.		17 623						19 373	

Tabulka 6: Testování řídících parametrů pro velikost populace 250 jedinců

Populace	F=0.2, Cr=0.2	F=0.2, Cr=0.6	F=0.2, Cr=0.8	F=0.4, Cr=0.2	F=0.4, Cr=0.8	F=0.6, Cr=0.2	F=0.6, Cr=0.8	F=0.8, Cr=0.2	F=0.8, Cr=0.8
1.	24 788	24 270	22 691	23 790	24 249	22 810	23 596	24 098	23 730
5.	22 595	21 479	21 032	22 484	21 796	22 656	22 043	22 930	22 600
10.	22 058	21 479	19 860	21 945	21 509	22 464	21 147	19 938	22 471
20.	21 321	20 949	18 842	21 518	21 193	21 573	21 147	19 938	22 436
30.	21 164	20 375	18 684	20 974	19 831	21 131	21 147	19 938	21 246
50.	21 164	19 890	18 630	20 637	19 222	21 131	20 270	19 938	20 659
80.	19 497	19 890	18 630	20 312	19 222	21 131	20 243	19 461	19 896
100.	19 497	19 630	18 630	20 312	19 222	21 131	19 044	19 387	19 896
130.	19 497	19 038	18 630	19 388	19 222	21 131	19 044	19 387	19 614
160.	19 410	19 038	18 630	19 388	19 222		19 044	19 387	19 614
190.	19 410	19 038		19 388			19 044	19 387	19 546
220.	19 410	19 038							19 546
250.	19 410								18 979
300.									18 979
350.									18 979
400.									

Tabulka 7: Testování řídících parametrů pro velikost populace 1000 jedinců

7.2 Testování dle optimalizačního kritéria

Výsledný rozvrh byl vygenerován s nastavením řídících parametrů, které v předešlé kapitole měly nejlepší výsledky. Rozvrhování operací bylo testováno pro velikost populace 500 a velikost populace 1 000. Nastavení ostatních řídících parametrů tedy bylo:

- Velikost populace: 500
- Počet populací: 1000
- F: 0.2
- Cr: 0.8

Každé optimalizační kritérium bylo testováno 3x pro stejné nastavení parametrů, aby jednotlivé výsledky mohly být vzájemně porovnány mezi sebou. Velikost testované datové sady zachycuje tabulka 8.

Počet zakázek	72
Počet operací	639
Počet strojů	160
Průměrný počet operací na zakázku	8,8
Průměrný počet možných strojů na operaci	6

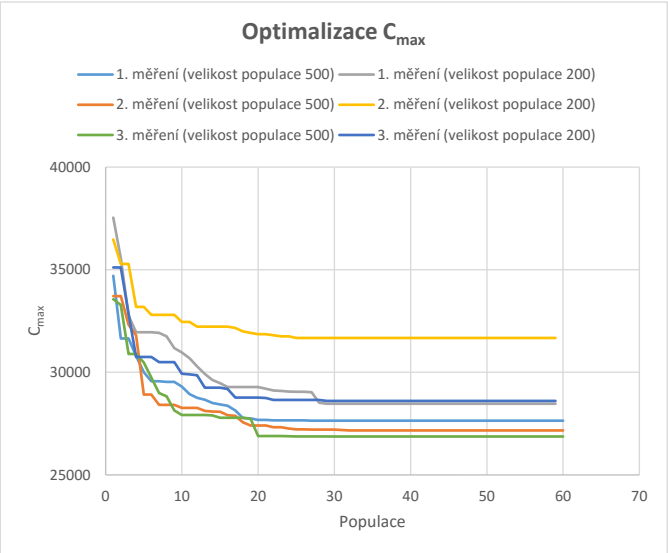
Tabulka 8: Velikost datové sady

7.2.1 Optimalizace dle maximální doby dokončení

Při výběru této optimalizace je pro výsledný výrobní rozvrh důležité, aby trval co nejkratší možnou dobu. Při výběru této optimalizace nelze příliš zohlednit termíny zakázek či jiné optimalizační kritéria. Průměrná doba výpočtu při velikosti populace 500 jedinců byla 15 minut, zatímco při velikosti populace 200 jedinců byla délka výpočtu cca 10 minut. Mezi těmito dvěma testy lze však zpozorovat i dle grafu 14 rozdíl ve kvalitě optimalizace. Test s více jedinci v populaci byl pomalejší. Jeho výsledky optimalizace však byly stabilnější a více optimalizované, než tomu bylo v případě testu s méně jedinci.

7.2.2 Priority zakázek

Tuto optimalizaci lze jinak nazvat jako celkovou váženou dobu dokončení všech zakázek, kde váha w je váha, která odpovídá výšce priority zakázky. Při této optimalizaci se evoluční proces snaží najít takový plán, kde budou prioritní zakázky plánované co nejvíce na začátek rozvrhu. Uvedené hodnoty v tabulce 10 jsou $\frac{1}{1000}$ původní hodnoty. U tohoto typu optimalizace jde vidět, že populace s větším počtem jedinců má stabilnější výsledky, než populace s menším počtem jedinců. V jednom případě byl výsledek populace s nižším počtem jedinců velice dobrý. Tento fakt



Obrázek 14: Graf znázorňující evoluci pro optimalizaci C_{max}

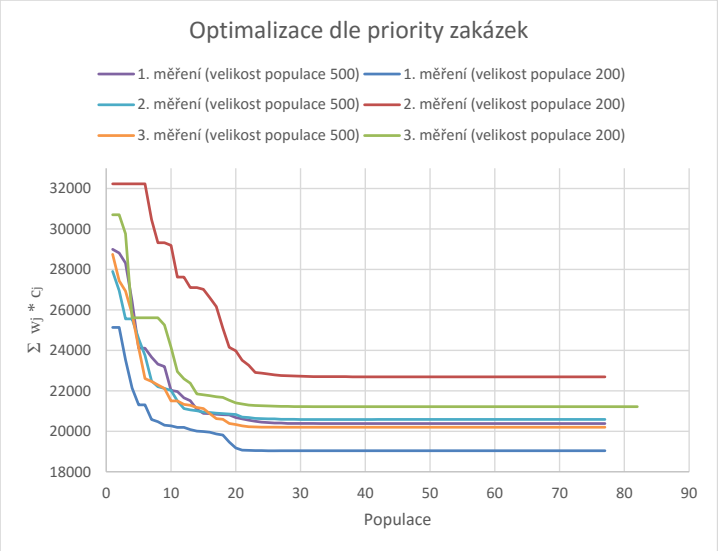
Populace	Počet jedinců: 500			Počet jedinců: 200		
	1.měření	2.měření	3.měření	1.měření	2.měření	3.měření
1.	34 704	33 714	33 562	37 537	36 471	35 110
2.	31 648	33 714	33 283	35 556	35 283	35 110
3.	31 648	32 323	30 897	32 764	35 283	32 863
4.	30 805	31 801	30 897	31 949	33 187	30 749
5.	30 000	28 913	30 471	31 949	33 187	29 934
10.	28 941	28 270	27 919	30 970	32 454	29 248
15.	28 371	28 080	27 781	29 475	31 858	28 767
20.	27 680	27 407	26 900	29 196	31 674	28 651
25.	27 656	27 217	26 878	28 469	31 674	28 606
30.	27 643	27 208	26 872	28 469	31 674	28 606
40.	27 643	27 167	26 872	28 469	31 674	28 606
50.	27 643	27 167	26 872	28 469	31 674	28 606
75.	-	27 167	-	-	-	-
100.	-	-	-	-	-	-
150.	-	-	-	-	-	-
200.	-	-	-	-	-	-

Tabulka 9: Optimalizace dle maximální doby dokončení C_{max}

však mohl být způsoben dobrou kvalitou jedinců hned v první náhodně vygenerované populaci, jelikož se tento dobrý výsledek v dalších dvou měření nepotvrdil.

Populace	Počet jedinců: 500			Počet jedinců: 200		
	1.měření	2.měření	3.měření	1.měření	2.měření	3.měření
1.	28 993	27 901	28 744	25 134	32 229	30 701
2.	28 815	26 955	27 435	25 134	32 229	30 701
3.	28 308	25 561	26 928	23 538	32 229	29 767
4.	26 420	25 561	25 810	22 137	32 229	25 611
5.	24 104	24 575	24 125	21 308	32 229	25 611
10.	22 025	22 017	21 503	20 273	29 191	24 153
15.	20 883	20 950	21 125	19 994	27 019	21 805
20.	20 683	20 831	20 336	19 180	23 975	21 404
25.	20 434	20 616	20 202	19 058	22 835	21 218
30.	20 384	20 588	20 202	19 040	22 721	21 218
40.	20 384	20 588	20 202	19 040	22 691	21 218
50.	20 384	20 587	20 202	19 040	22 691	21 218
75.	20 384	20 587	20 202	19 040	22 691	-
100.	-	20 587	-	-	-	-
150.	-	-	-	-	-	-
200.	-	-	-	-	-	-

Tabulka 10: Optimalizace dle priorit zakázek



Obrázek 15: Graf znázorňující evoluci pro optimalizaci prioritních zakázek

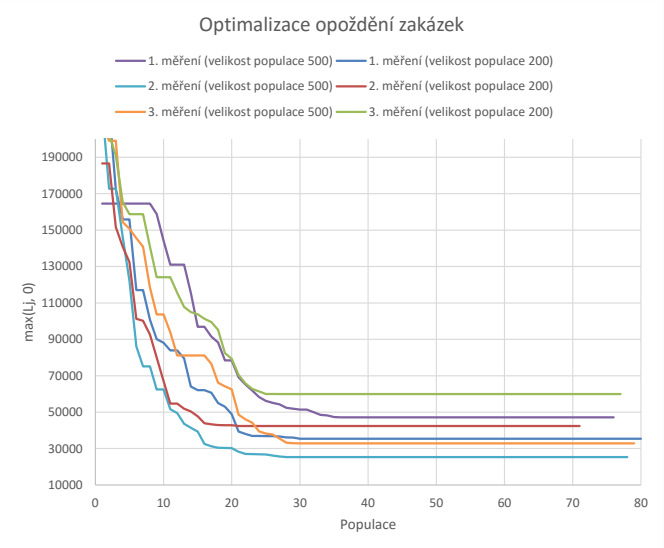
7.2.3 Maximální opoždění

Maximální opoždění je optimalizace vytvářející takový plán, kde zakázky nebudou mít opoždění oproti termínu dodání. Tato hodnota může nabývat pouze kladných hodnot. Pokud je hodnota

při optimalizaci rovná nule, je jasné, že se úspěšně podařilo sestavit plán, ve kterém není žádná zakázka dokončena se zpožděním. Výsledky testování jsou v tabulce 11 a zobrazené v grafu 16

Populace	Počet jedinců: 500			Počet jedinců: 200		
	1.měření	2.měření	3.měření	1.měření	2.měření	3.měření
1.	164 555	217 564	207 343	255 964	186 600	261 807
2.	164 555	172 716	199 086	217 063	186 600	202 508
3.	164 555	172 716	199 086	172 446	151 681	191 034
4.	164 555	146 308	154 361	155 867	140 927	165 450
5.	164 555	122 272	150 759	155 867	132 434	158 705
10.	144 206	62 494	103 661	88 115	67 104	124 108
15.	96 862	39 371	81 174	62 038	47 668	103 746
20.	78 426	30 205	62 607	48 878	42 815	79 354
25.	56 169	26 776	38 182	36 880	42 387	59 998
30.	51 365	25 318	32 858	35 397	42 387	59 944
40.	47 147	25 318	32 855	35 397	42 387	59 944
50.	47 147	25 318	32 855	35 397	42 387	59 944
75.	47 147	25 318	32 855	35 397	-	59944
100.	-	-	-	-	-	
150.	-	-	-	-	-	

Tabulka 11: Optimalizace maximálního opoždění zakázek



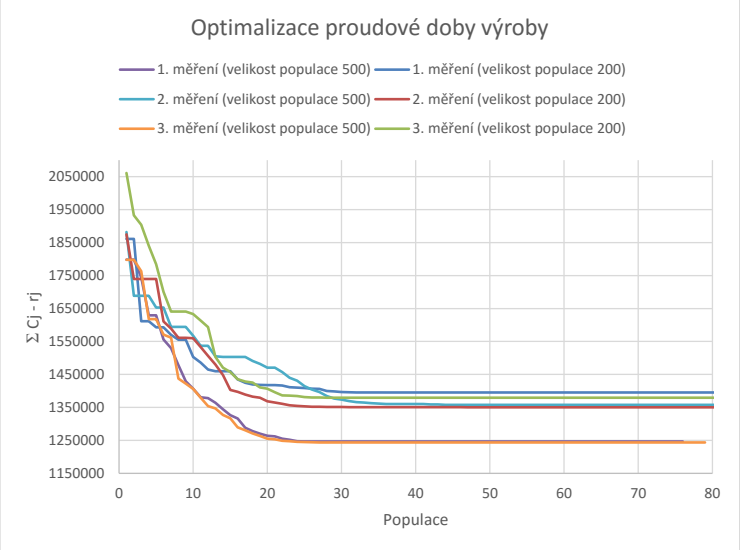
Obrázek 16: Graf znázorňující evoluci pro optimalizaci opoždění zakázek

7.2.4 Celková proudová doba výroby

Optimalizací podle celkové proudové doby výroby znamená, že pokud je nějaká zakázka uvolněna do výroby (může být zahájena její první operace), pak se tato optimalizace snaží co nejvíce zkrátit časový interval mezi dokončením zakázky a uvolněním zakázky do výroby. Naměřené hodnoty z tabulky 12 jsou vidět v grafu 17. I při této optimalizaci bylo dokázáno, že při vyšším počtu jedinců bylo dosaženo lepší optimalizace.

Populace	Počet jedinců: 500			Počet jedinců: 200		
	1.měření	2.měření	3.měření	1.měření	2.měření	3.měření
1.	1798272	1882025	1797706	1861071	1874776	2060999
2.	1798272	1688880	1797706	1861071	1739395	1933035
3.	1744819	1688880	1762913	1611600	1739395	1903961
4.	1629441	1688880	1617973	1611600	1739395	1840226
5.	1629441	1653083	1617973	1592944	1739395	1783995
10.	1406850	1567426	1406370	1503295	1559833	1632983
15.	1326346	1502934	1317105	1459537	1402804	1457079
20.	1264144	1470315	1255396	1417444	1368593	1406959
25.	1246736	1414294	1245245	1408535	1353070	1381354
30.	1246688	1373487	1243725	1396408	1351588	1379486
40.	1246688	1360132	1243725	1395307	1350690	1379449
50.	1246688	1357769	1243725	1395307	1350313	1379449
75.	1246688	1357769	1243725	1395307	1350313	1379449
100.	-	-	-	-	-	
150.	-	-	-	-	-	

Tabulka 12: Optimalizace dle priorit zakázek



Obrázek 17: Graf znázorňující evoluci pro optimalizaci vážené proudové doby výroby

7.3 Měření kvality a výkonu z hlediska velikosti datové sady

Měření algoritmu a porovnávání kvality výsledného rozvrhu je realizováno na optimalizačním kritériu C_{max}. Jsou mezi sebou porovnávány dvě datové sady. První datová sada má velikost uvedenou v tabulce 8 a druhá datová sada je uvedená v tabulce 4. Velikost populace je nastavena na 500 jedinců. Tabulka 13 vystihuje výsledky porovnání různě velkých datovou sad při optimalizaci hodnoty C_{max}.

	1.datová sada	2.datová sada
Hodnota C_{max}	29 134	19 062
Doba trvání optimalizace v minutách	13	2
Ukončení v populaci	46	60

Tabulka 13: Optimalizace C_{max} s různě velkou datovou sadou

Bylo zjištěno, že aby algoritmus našel optimalizovaný rozvrh pro větší datovou sadu (639 operací), musel běžet významně déle než při optimalizaci pro menší datovou sadu (144 operací). Výhodnější tedy je, aby rozvrhování bylo prováděno v menších časových horizontech. To přináší nejen výhodu, že výsledný rozvrh je vygenerován v jednotkách minut, ale také je větší pravděpodobnost, že optimalizace bude kvalitnější. Při optimalizaci větší datové sady bylo také vypozorováno, že evoluce projde menším počtem populací, než je tomu při optimalizaci větší datové sady. Tento fakt mohlo způsobit to, že pro vysoký počet operací byl nízký počet jedinců v populaci. To znamená, že populace mohla být málo „rozmanitá“ a brzy uvízla v některém z lokálního minima. Samotné hodnoty C_{max} mezi sebou samozřejmě porovnávat nelze, jelikož se jedná o naprosto rozdílný počet zakázek. Ukázka výsledného rozvrhu a uživatelského rozhraní po optimalizaci C_{max} u 2. datové sady je v příloze B.1.

8 Závěr

Cílem této diplomové práce bylo vytvořit modul kapacitního plánování pro systém iMPROVE iT!.

V teoretické části práce byl obecně vysvětlen pojem kapacitního plánování, historický vývoj systémů kapacitního plánování i rozdíly mezi systémy řešícími tuto problematiku. Dále se práce věnovala základním pojmům v oblasti kapacitního plánování. Byly popsány rozdíly mezi plánováním a rozvrhováním, statické parametry, dynamické parametry a omezující podmínky tohoto problému. Následně byla představena Grahamova klasifikace s popisem všech jejích parametrů, různé multi-operační problémy, typy rozvrhů a výpočetní složitost problému rozvrhování. Na tuto kapitolu navazovaly samotné možnosti řešení problematiky kapacitního plánování. Čtenář byl seznámen s jednou exaktní metodou, konkrétně se jednalo o metodu větví a mezí. Jako další varianty byly představeny nejpoužívanější heuristické metody - simulované žíhání, prioritní pravidla, algoritmy založené na evoluci. Lehce byly nastíněny algoritmy s prvky umělé inteligence.

Kapitola 5 načíná praktickou část práce věnující se seznámením s konkrétním byznys problémem. Jako algoritmus pro řešení tohoto byznys problému byl vybrán algoritmus diferenciální evoluce. Tento algoritmus je nejdříve popsán obecně a poté z hlediska použití pro daný byznys problém. Výsledkem diferenciální evoluce je optimalizovaný výrobní rozvrh, uložený ve formě disjunktivního grafu. Tento graf a operace s ním spojené, jako jsou topologické uspořádání a nalezení kritické cesty, jsou popsány textově a jejich implementace je popsána pseudokódem. Pro ohodnocení rozvrhu jsou představeny čtyři vybrané účelové funkce.

Kapitola 6 se týkala samotné integrace modulu do systému iMPROVE iT!. V této kapitole byl ve zkratce popsán systém iMPROVE iT! a byly definovány vztahy s ostatními moduly tohoto systému. Nejdůležitější částí integrace bylo porozumění modulu „ImproveIT.Operations“, který má struktury založené na normě ANSI/ISA 95. Tato norma byla ve zkratce představena. Aby bylo možné data ze struktur založených na této normě korektně převádět do struktur modulu kapacitního plánování i zpět, bylo stěžejní pečlivě tuto normu nastudovat. Poslední část této kapitoly je věnována popisu druhého modulu, který slouží k zobrazení uživatelského rozhraní ve formě Ganttova grafu a obsahuje funkčnost pro jeho ovládání.

V poslední kapitole 7 probíhá testování na reálných datech. V první části testování jsou testovány různé vstupní parametry algoritmu diferenciální evoluce a následně je vybrána nejlepší konfigurace těchto parametrů. Pro nejlepší konfiguraci vstupních parametrů jsou následně testovány a porovnávány všechny čtyři typy optimalizace rozvrhu pro dvě různé velikosti populace.

Výstupem práce jsou výrobní rozvrhy ve formě Ganttova grafu dle vybraného optimalizačního kritéria. Nutno říci, že nelze pevně stanovit přesný počet vstupních zakázek algoritmu, tak aby výsledný rozvrh byl opravdu maximálně optimalizovaný. Pokud však určitý výrobní podnik chce mít optimalizovaný rozvrh v jednotkách minut, je lepším řešením rozvrhovat menší počet zakázek, například týkajících se následujících několika směn. Pak je pravděpodobnost, že evoluční algoritmus uvízne v některém lokálním minimu menší, než kdyby byl vstupem algoritmu

obrovský počet zakázek. Výhodou modulu pro kapacitní plánování bez pochyb je schopnost vygenerovat optimalizovaný rozvrh během pár minut. Pokud by takový rozvrh měl vytvořit člověk, stráví nad tímto problémem bez diskuze v lepším případě minimálně několik hodin.

Cílem této práce bylo vytvořit modul pro kapacitní plánování. Jako typ plánování byla zvolena technika optimalizace rozvrhování dle čtyř různých kritérií. Jako kapacity byly brány v potaz kapacity strojů. Při rozvrhování byly dodrženy precedenční omezení operací u zakázek.

Tato práce by v budoucnu mohla být rozšířena například o implementaci algoritmu s prvky umělé inteligence, implementací dalších účelových funkcí nebo by mohly být brány v potaz další kapacitní omezení např. personál.

Literatura

[1] Velkoborský, J., Roub, *Aplikace APS představují převrat v řízení výroby,* Computerworld 36/1999

[2] František Manlig, František Koblasa *Plánování a rozvrhování výroby (vybrané kapitoly) ,* Technická univerzita v Liberci, 2015

[3] Y. Yuan, H. Xu, *Flexible job shop scheduling using hybrid differential evolution algorithms,* Computers & industrial engineering, v.65 n.2, p.246-260, June, 2013

[4] Peter Brucker, Bernd Jurisch, Bernd Sievers *A branch and bound algorithm for the job-shop scheduling problem,* Discrete Applied Mathematics, v.49 n.1-3, p.107-127, March, 1994

[5] ISAEUROPE *About ISA Europe ,* [online]. 17.11.2014 [cit. 2016-03-21]. Dostupné z: `http://isaeurope.org/isa-announces-2014-fellows-celebrating-excellence-award-honorees/`

[6] ANSI/ISA-95.00.01-2010 *Enterprise-control System Integration - Part 1: Models and Terminology ,* Park, N. C: ISA, 2010. ISBN 9781936007479

[7] Natallia Kokash *An introduction to heuristic algorithms ,* University of Trento, Italy

[8] ZELINKA, Ivan *Evoluční výpočetní techniky: principy a aplikace. ,* 1. vyd. Praha: BEN - technická literatura, 2009. ISBN 978-80-7300-218-3

[9] BTAZEWICZ, Jacek, Wolfgang DOMSCHKE a Erwin PESCH *The job shop scheduling problem: Conventional and new solution techniques. ,* European Journal of Operational Research. Elsevier, 1996, 1-33

[10] Pawel Piotr Lukaszewicz *METAHEURISTICS FOR JOB SHOP SCHEDULING PROBLEM, COMPARISON OF EFFECTIVE METHODS ,* Aarhus School of Business, Master Thesis – December 2005

[11] Peter Brucker *Scheduling algorithms. 5th ed. ,* New York: Springer, 2007. ISBN 9783540695158.

[12] KENNETH V. PRICE, RAINER M. STORN a JOUNI A. LAMPINEN *Differential Evolution: A Practical Approach to Global Optimization ,* Lappeenranta University of Technology: Springer, 1998. ISBN 10 3-540-20950-6..

[13] ANSI/ISA-95.00.02-2010 *Enterprise-control System Integration - Part 2: Object Model Attributes ,* Park, N. C: ISA, 2010. ISBN 9781936007486

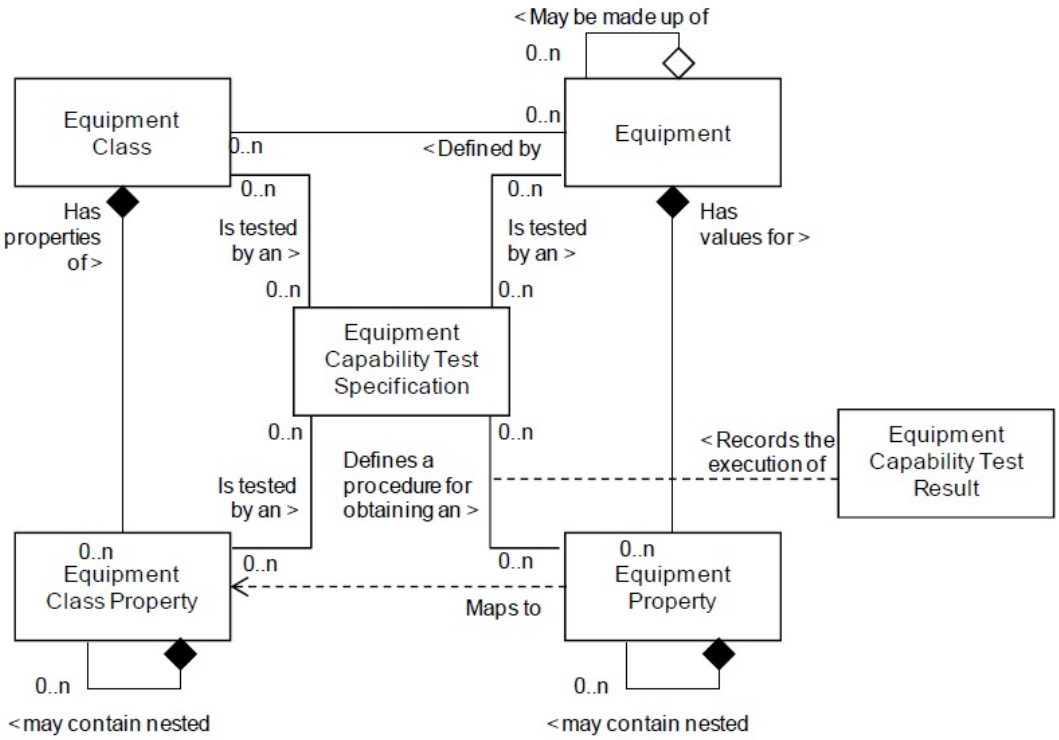
[14] Wenyin Gong, Zhihua Cai, and Charles X. Ling *DE/BBO: A Hybrid Differential Evolution with Biogeography-Based Optimization for Global Numerical Optimization ,* Soft Computing, v.15 n.4, p.645-665, March, 2010

[15] CHATZICHARALAMPOUS, Stamatia. Logistics and Planning Professionals Increasing the Degree of Sophistication in your Supply Chain Planning. *AIMMS Supply Chain Blog ,* [online]. 2013 [cit. 2016-04-27]. Dostupné z: http://supplychainblog.aimms.com/2013/11/increasing-the-degree-of-sophistication-in-your-supply-chain-planning/

[16] PAVEL VRBA a VLADIMÍR MAŘÍK, *Software (nejen) pro Airbus / Industrie 4.0 ,* TecniCall [online]. 2015 [cit. 2016-04-27]. Dostupné z: http://www.tecnicall.cz/clanek/2015-02-airbus

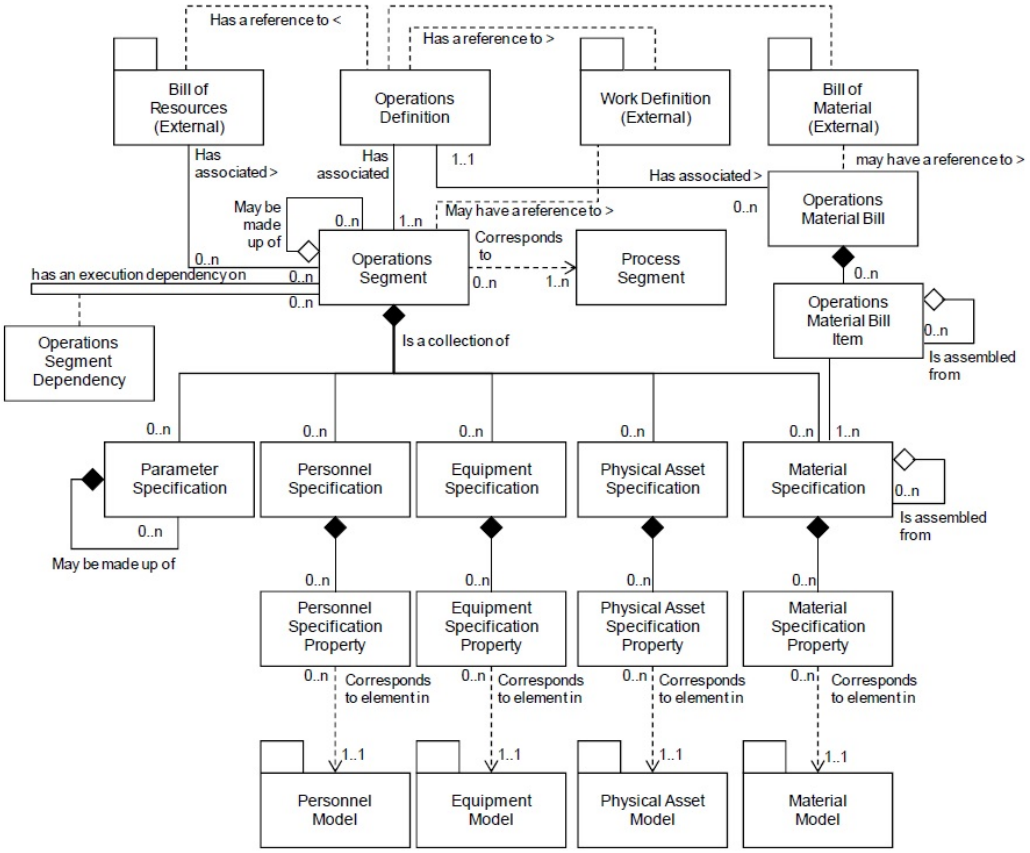
A ANSI/ISA 95 modely

A.1 Model zařízení



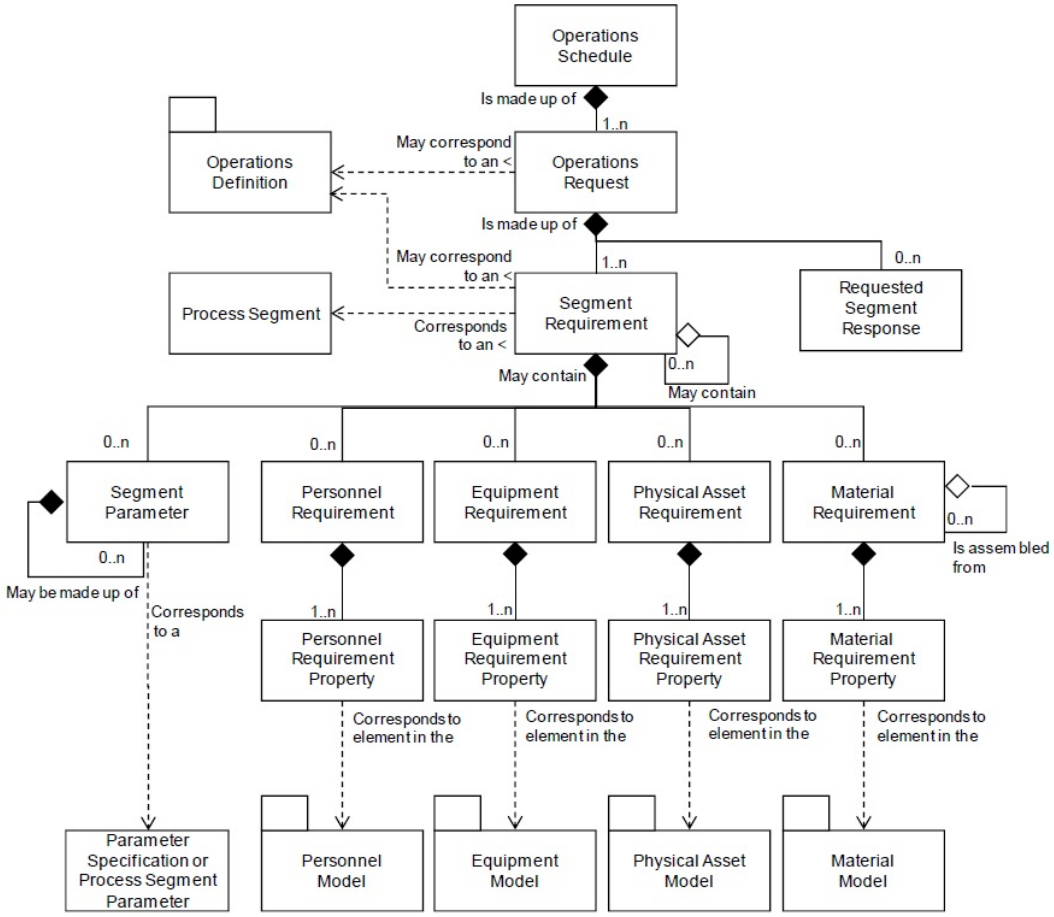
Obrázek 18: Model nástrojů, převzato z [13]

A.2 Model definice operací



Obrázek 19: Model definic operací, převzato z [13]

A.3 Model rozvrhování operací



Obrázek 20: Model rozvrhování operací, převzato z [13]

B Ukázky uživatelského rozhraní

B.1 Ukázka výrobního rozvrhu po optimalizaci Cmax